Technische Universität Dresden

Fakultät Informatik
Institut für Systemarchitektur
Professur Rechnernetze
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Medieninformatiker

Eine Middleware für die nahtlose Integration und Kooperation von Geräten

Roman Wilkowski
(Geboren am 8. Oktober 1979 in Berlin)

Betreuer: Herr Dr.-Ing. Waltenegus Dargie

Dresden, 21. Januar 2008

Aufgabenstellung

Ziel:
Im Umfeld des Ubiquitous Computing sind Gegenstände so intelligent, dass sie sich der Aufmerksamkeit des Benutzers entziehen und den Menschen bei seinen Tätigkeiten unbemerkt unterstützen. Der Benutzer interagiert mühelos und ohne sich dessen bewusst zu sein mit einer Vielzahl an unterschiedlichen Geräten (mobil, eingebettet, stationär) und kann so seine Aufmerksamkeit auf die wesentlichen Aufgaben richten. Ubiquitous Computing setzt die dynamische Integration und Kooperation von Geräten voraus.

Durch die dynamische und nahtlose Integration von Geräten und Ressourcen wird der Umgang mit der Technik intuitiver und in Zusammenarbeit mit Kontextbewusstsein (Context-Awareness) kann so, insbesondere älteren Menschen, ein unabhängigeres Leben ermöglicht werden.

Das Ziel dieser Diplomarbeit ist es, ausgewählte Konzepte des Ubiquitous Computing und der Context-Awareness zu untersuchen. Dabei soll deren Umsetzung besonders älteren Menschen in ihrer Wohnumgebung und deren alltäglichen Leben entscheidende Hilfestellung leisten können. Die Arbeit soll typische Situationen älterer Menschen, wie Notfälle, Vergesslichkeit und Verzweiflung im Umgang mit fehlerhaften Betriebszuständen von Geräten und Anwendungen veranschaulichen.

Bearbeitungsschwerpunkte:

1. Auseinandersetzung mit verwandten wissenschaftlichen Arbeiten (Smart Space, Kontextbewusste Infrastrukturen, Ontologien)
2. Identifikation von Anwendungsbereichen in der Wohnumgebung
3. Entwurf einer konzeptionellen Architektur
4. Untersuchung des Einsatzes von Ontologien (OWL) zur Modellierung von Smart Environments
5. Regeln und Policen für die nahtlose Kooperation und Integration von Geräten
6. Erstellung eines Prototyps

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Informatik eingereichte Diplomarbeit zum Thema:

Eine Middleware für die nahtlose Integration und Kooperation von Geräten

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Dresden, den 21. Januar 2008

Roman Wilkowski

Abstract

In the ideal Ubiquitous Computing environment, devices and resources are intelligent enough to conceal themselves from the attention of the user with the goal to support him. Therefore it is required, that those intelligent devices cooperate with each other. The user shall not be aware of this interaction but concentrate fully on his tasks. Consequently, dynamic device integration and cooperation is essential. We achieve this by proposing a middleware and a three-tier cooperation architecture, defining policies, rules and tasks. Devices and services are described abstractly with ontologies (OWL). The cooperation between devices is defined by rules. A protype based on OSGi presents the feasibility of the here introduced approach.

Kurzfassung

Ubiquitous Computing steht für die Allgegenwärtigkeit von Rechnern, welche die Aufgabe haben die Menschen zu unterstützen. Dazu ist es erforderlich, dass die räumlich verteilte intelligente Technik miteinander kooperiert. Diese Vorgänge sollen sich der Aufmerksamkeit des Menschen entziehen, damit er sich auf seine wesentlichen Aufgaben konzentrieren kann. Voraussetzung dafür ist die dynamische Integration und Kooperation von Geräten. In dieser Arbeit wird eine Middleware entwickelt, die dies ermöglichen soll. Es wird eine dreischichtige Kooperationsarchitektur eingeführt. Sie definiert die Policen auf der obersten, Regeln auf der mittleren und Tasks auf der untersten Schicht. Geräte und Dienste werden mit Ontologien (OWL) abstrakt beschrieben, und anhand von Regeln die Kooperation zwischen Geräten festgelegt. Eine prototypische Umsetzung in OSGi zeigt die Machbarkeit des vorgestellten Ansatzes.

Inhaltsverzeichnis

1 Einleitung **5**
1.1 Motivation . . . 5
1.2 Problemstellung . . . 6
1.3 Lösungsansatz . . . 6
1.4 Struktur der Arbeit . . . 7

2 Identifikation der Anwendungsdomäne **9**
2.1 Szenarien . . . 9
2.2 Anwendungsfälle . . . 12
2.3 Anforderungsanalyse . . . 18
2.3.1 Funktionale Anforderungen . . . 18
2.3.2 Nichtfunktionale Anforderungen . . . 24
2.4 Zusammenfassung . . . 25

3 Verwandte Arbeiten und existierende Technologien **27**
3.1 Ontologien und Regeln . . . 27
3.1.1 OWL . . . 27
3.1.2 Semantic Web Rule Language (SWRL) . . . 28
3.1.3 Context-Awareness . . . 28
3.1.4 Gu's Ontology-based Context Model for Intelligent Environments . . . 28
3.1.5 Chen's Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) . . . 29
3.2 Verwandte Ansätze . . . 31
3.2.1 Interplay . . . 32
3.2.2 AMIGO . . . 35
3.2.3 Nahtlose Integration von Ausgabegeräten . . . 37
3.2.4 Dienstorientierte kontextbewusste Middleware . . . 38
3.2.5 DAFNE . . . 41
3.2.6 Schlussfolgern auf Kontext im Smarthome: Ein SWRL Ansatz . . . 42
3.3 Heimvernetzungstechnologien . . . 44
3.4 OSGi . . . 45
3.5 Dienste . . . 46
3.6 Zusammenfassung . . . 47

4 Konzeption **49**
4.1 Regeln und Policen . . . 49
4.1.1 Policen . . . 50
4.1.2 Regeln . . . 53
4.1.3 Tasks . . . 54
4.2 Architektur . . . 54
4.2.1 Home Controller . . . 56

4.2.2 Residential Gateway . . . 60
4.2.3 Knowledge Base . . . 60
4.2.4 Device Cooperation Manager . . . 61
4.2.5 Multimodaler Benutzerschnittstellen Manager . . . 62
4.2.6 User Modeling & User Profiling . . . 63
4.2.7 Policy Interface . . . 64
4.2.8 Policy Manager . . . 64
4.2.9 Policy-to-Rule Translator . . . 64
4.2.10 Regelschicht . . . 64
4.2.10.1 Rule Engine . . . 64
4.2.10.2 Regel Monitor . . . 64
4.2.10.3 Regel Konfliktmanager . . . 65
4.2.10.4 Regel Entdecker . . . 65
4.2.10.5 Reasoner . . . 65
4.2.10.6 Regelbasis . . . 65
4.2.10.7 Regelübersetzer . . . 66
4.2.10.8 Regelersteller . . . 66
4.2.10.9 Regel Inspektor . . . 66
4.2.10.10 Regel Debugger . . . 67
4.2.11 Taskmodellierung . . . 67
4.2.12 Ereignis Manager . . . 68
4.2.13 Context Service . . . 70
4.2.14 Device Registry & Classification . . . 71
4.2.14.1 Device Registry . . . 71
4.2.14.2 Device Classification . . . 72
4.2.14.3 Semantic Service Discovery . . . 73
4.2.15 Privatssphären Manager . . . 73
4.2.16 Netzwerkabstraktionsschicht . . . 73
4.2.17 Plattform . . . 74
4.3 Ontologien . . . 74
4.3.1 Domänen-Ontologie . . . 74
4.3.2 Geräteontologie . . . 75
4.3.3 Nutzerontologie . . . 75
4.3.4 Taskontologie . . . 75
4.4 Entscheidungsbäume . . . 75
4.5 Zusammenfassung . . . 77

5 Implementierung **79**
5.1 Klassifikation von Geräten und Ereignissen . . . 79
5.1.1 Repräsentation der Konzepte in einer Domänenontologie . . . 79
5.1.2 Abbilden von OWL-Konzepten auf Dienste . . . 82
5.1.3 Kooperation von Geräten . . . 82
5.1.4 Integration von existierenden Geräten . . . 83
5.1.5 Integration neuer Geräte . . . 83
5.1.6 Dienste . . . 83
5.2 Regelschicht . . . 85
5.3 Task . . . 86
5.4 Geräte als OSGi-Bundles . . . 88

5.4.1 Ereignisbehandlung 88
5.4.2 Kontextbenachrichtigung 88

6 Bewertung der Arbeit **89**

7 Zusammenfassung **91**

Literaturverzeichnis **93**

Abbildungsverzeichnis **97**

1 Einleitung

1.1 Motivation

Ubiquitous Computing (Ubicomp) steht für die Allgegenwärtigkeit von Rechnern. In der idealen Ubiquitous Computing Umgebung ist der Nutzer von intelligenter Technik umgeben, ohne diese bewusst wahrzunehmen. Die einzelnen Geräte kooperieren miteinander, gehen auf vorhandenen Kontext ein und vereinfachen so den Alltag des Nutzers. Es bedeutet einen Innovationssprung von der Zeit in der der Nutzer dem Computer und den Geräten Befehle geben und diese bedienen musste, bis hin zu einer Zeit, in der Computer und andere Geräte Aufgaben selbständig erkennen und ausführen, ohne den Benutzer in seinen Aktivitäten zu unterbrechen.

In der Publikation „Challenges: An Application Model for Pervasive Computing“ [2] werden folgende Grundsätze aufgestellt:

1. Ein Gerät stellt eine Schnittstelle zur Anwendung dar, beziehungsweise zu den Daten, und ist nicht eine Ansammlung von Software.
2. Eine Anwendung ist für den Nutzer das Mittel um eine Aufgabe auszuführen, nicht eine Software, die verfügbare Ressourcen verbraucht und dem Selbstzweck dient.
3. Die Computing Environment erweitert die Umgebung des Nutzers durch zusätzliche Informationen und soll nicht ein virtueller Raum für das Speichern und Ausführen von Software sein.

Die unbewusste Interaktion mit der Technik zur Konzentration auf die wesentlichen Aufgaben erfordert, dass sich die zur Verfügung stehende Technik und die Geräte unauffällig in die Umgebung des Nutzers integrieren und miteinander kooperieren. Der Nutzer soll die Anwesenheit eines solchen Systems primär durch dessen Unterstützungsleistung wahrnehmen. Es wird untersucht, ob sich das Leben des Nutzers wirklich vereinfacht, oder durch die Bedienung unnötige Anforderungen gestellt werden. In dieser Arbeit wird der Schwerpunkt insbesondere auf die Lebensqualität und die Wohnsituation älterer Menschen gelegt. Sie soll sich durch den Einsatz elektronischer Hilfen signifikant verbessern und ihnen einen Teil ihrer Unabhängigkeit bewahren.

Die Motivation für Smarthomes

Smarthomes sind intelligente Wohnumgebungen, in denen versucht wird die Ideen des Ubiquitous Computing umzusetzen. Durch den Einsatz allgegenwärtiger Technik wird den Menschen das Leben erleichtert. Besonders ältere Menschen und Menschen mit Behinderungen haben die größten Vorteile von Smarthomes mit kontextbewusster Technik und Anwendungen. Für sie bedeutet der Einsatz dieser Technologien mehr Unabhängigkeit [21].

1.2 Problemstellung

In einem Smart Home werden existierende Geräte dynamisch in ein Gesamtsystem integriert, in dem sie miteinander kooperieren um besonders älteren Menschen ein unabhängigeres und angenehmeres Leben zu ermöglichen. Diese Integration erfolgt nahtlos, also ohne dass der Nutzer diese Geräte konfigurieren oder Einstellungen ändern muss. Die Geräte melden sich dynamisch an und stehen dann mit allen Funktionalitäten zur Verfügung. Die zu entwickelnde Middleware soll kontextabhängig sein, den Nutzer, seine Umgebung und seine vorhandenen Geräte berücksichtigen. Die Geräte kooperieren selbstständig. So kann sich der Nutzer auf die Erledigung seiner Aufgaben und Ziele konzentrieren, also auf das *„Was will ich machen?“* und muss nicht darüber nachdenken *„Wie muss ich die Geräte bedienen?“* um ein Ziel zu erreichen.

Es gibt Hardware-Standards wie UPnP, die das Anschließen und die Kommunikation zwischen UPnP-Geräten spezifizieren. Dazu müssen alle Geräte diesen Standard implementieren. Die mögliche Kommunikation zwischen den verschiedenen Geräteklassen wird im Vorhinein festgelegt. Es ist somit kein dynamisches und intelligentes Verhalten adhoc möglich. Die möglichen Szenarien müssen im Vorfeld definiert werden.
In dieser Arbeit soll eine Middleware entwickelt werden, die verschiedenste Geräte integriert. Dazu wird auf die folgenden Fragestellungen eingegangen. Wie können heterogene Geräte miteinander kooperieren? Welche Regeln und Policen sind dafür nötig?

1.3 Lösungsansatz

Damit sich heterogene Geräte verschiedener Herstellern verständigen können, wird eine abstrakte Sprache zur Beschreibung der Geräte, ihrer Funktionalitäten und Dienste benötigt. Dafür eignet sich die Web Ontology Language (OWL). Mit diesem ontologie-basierten Ansatz können die Geräte, Regeln und Policen formal und maschinenlesbar spezifiziert werden. Die einzelnen Geräte die im Haushalt vorkommen, werden abstrakt in einer Ontologie beschrieben und als Dienste umgesetzt. Die Geräte sind vollständig in die Middleware integriert, nachdem sie sich mit einer Selbstbeschreibung ihrer Fähigkeiten und Dienste angemeldet haben. Zur Umsetzung dient eine serviceorientierte Architektur (SOA). Das System benötigt Kenntnisse über die Präferenzen und die aktuelle Situation des Nutzers. Hierfür ist die Erfassung des Kontextes und das logische Schlussfolgern auf dem Kontext entscheidend. Diese Aufgabe wird von einem Kontextdienst übernommen.

Ein interessanter Aspekt, der über die Aufgabenstellung dieser Arbeit hinausgeht, ist unter anderem die Fragestellung, wie sich die Funktionalitäten der vernetzten Geräte dynamisch in die Benutzerschnittstelle integrieren lassen, insbesondere bisher unbekannte Gerätefunktionalitäten.

Diese Arbeit soll über die reine Vernetzung von Geräten, in vordefinierten Szenarien und Abläufen, hinausgehen. Es müssen spezielle Regeln für Standardsituationen und Szenarien in der Wohnumgebung vordefiniert werden. Die Middleware soll Entscheidungen für den Nutzer übernehmen oder dem Nutzer Handlungsoptionen in leicht verständlicher Weise anbieten.

1.4 Struktur der Arbeit

Um genau zu verstehen wie die Anforderungen an eine solche Middleware sein sollen wird in Kapitel 2 die Anwendungsdomäne genauer analysiert. Es werden typische Anwendungsfälle dargestellt, in denen die Kooperation von Geräten dem Nutzer hilft und sein Leben erleichtert. Aus diesen spezifischen Szenarien werden Anwendungsfälle ermittelt, aus denen sich die Anforderungen bestimmen lassen. In Kapitel 3 werden existierende Technologien und verwandte Arbeiten gesichtet. Diese liefern die Ideen für die Konzeption in Kapitel 4. Die Beschreibung des Prototyps zur Konzeption folgt in Kapitel 5. Die Verifikation meines Ansatzes in Kapitel 6 bewertet die Vorgehensweise. Die Zusammenfassung in Kapitel 7 schließt diese Diplomarbeit mit einer Bewertung und einem Ausblick ab.

2 Identifikation der Anwendungsdomäne

Das Ziel dieser Arbeit ist es, Konzepte des Ubiquitous Computing und der Context-Awareness zu untersuchen und ob die Umsetzung dieser Konzepte älteren Menschen in ihrem alltäglichen Leben entscheidende Hilfestellung leisten und ihnen so ein unabhängigeres Leben ermöglichen kann. Der Alltag insbesondere älterer Menschen, spielt sich größtenteils in ihrer Wohnumgebung ab. Es sollen daher Anwendungsbereiche in der Wohnumgebung identifiziert werden, in denen die zu entwickelnde Middleware-Lösung diesen Menschen Hilfestellung leisten kann. Um die Anforderungen an eine solche Middleware zu ermitteln, werden nachfolgend Szenarien dargestellt, die ausführlich die notwendigen und interessanten Aspekte dieses Anwendungsbereichs betrachten. Daraus werden die Anforderungen (Requirements) ermittelt, die generell an ein solches System gestellt werden können. Die Konzeption der zu entwickelnden Middleware basiert auf diesen Anforderungen und wird in Kapitel 4 eingehend beschrieben.

Typische Situationen älterer Menschen, die alleine leben sind:

- Notfälle
- Vergesslichkeit
- Hilflosigkeit im Umgang mit fehlerhaften Geräten und Anwendungen
- Schwierigkeiten bei der Umsetzung der existierenden Technik
- Nicht ausreichende extrinsische Anreize (durch Arbeitsalltag, Nachbarschaft, Familie, politische Aktivitäten usw.)

2.1 Szenarien

Für die Analyse der Anforderungen werden nachfolgende Szenarien zu Grunde gelegt, in denen die unterschiedlichen Anwendungsbereiche herausgestellt werden. Hauptperson dieser Szenarien ist die ältere Frau Janet, die allein zu Hause wohnt. Sie hat gerade ihren 70igsten Geburtstag gefeiert und von ihrer Familie moderne Haushaltsgeräte, Kommunikationsgeräte und Unterhaltungselektronik geschenkt bekommen oder sich selber angeschafft, die ihr helfen sollen ihren Alltag zu vereinfachen. Im alltäglichen Umgang mit diesen Geräten können sich unterschiedliche Szenarien ergeben, in denen Janet Geräte bedient und verschiedene Geräte kooperieren um Janet zu helfen und ihr den Alltag zu erleichtern.

Szenario 1

Szenario: Anzeige einer Fehlermeldung auf einem beliebigen Gerät, welches die Aufmerksamkeit von Janet hat

Akteur-Instanzen:

- Janet: Bewohnerin
- Gerätehersteller
- Waschmaschine, Geschirrspüler, Fernseher: Geräte im Haushalt

Ereignissfluss:

1. Janet bedient gerade die Waschmaschine im Keller. Auf dem Display der Waschmaschine wird eine kurze Fehlermeldung angezeigt, dass die Geschirrspülmaschine in der Küche läuft, aber kein Wasser auspumpt, und das aktuelle Reinigungsprogramm gestoppt wurde. Es wird angezeigt, dass dieser Fehler eine Reaktion von ihr erfordert. Normalerweise würde Janet jetzt im Benutzerhandbuch nachlesen, wie ein solcher Fehler behoben werden kann. Sie weiß, dass ihr das System die benötigte Hilfestellung anbietet. Aber ohne Brille kann Janet das kleine Display der Waschmaschine nicht gut erkennen, und sie möchte sich auch lieber bequem hinsetzen, während sie sich den Lösungsvorschlag zu dem Fehler der Geschirrspülmaschine anzeigen lässt.

2. Als Janet in das Wohnzimmer kommt, wird die Fehlermitteilung der Geschirrspülmaschine und die dazugehörige Hilfestellung bereits auf dem großen Fernseher angezeigt. Mit dem Fernseher kann Janet besser interagieren, da sie dazu die Fernbedienung nutzen kann, mit der sie sehr vertraut ist (bessere Interaktionsmöglichkeit) und auf dem Fernseher eine große übersichtliche Menüführung mit Bildern und Sprache dargestellt werden kann.

3. Die Hilfestellung empfiehlt bei diesem Problem, den Hersteller zu kontaktieren um weitere Hilfe anzufordern.

4. Janet wählt auf dem Menü den Punkt *Hilfe vom Hersteller anfordern* aus. Das System stellt eine Internetverbindung zum Gerätehersteller her und sendet den Fehlerbericht an ihn.

5. Auf Herstellerseite wird zu dem Fehler eine Fehlerlösung aus einer Datenbank herausgesucht und an das System geschickt.

6. Das System stellt fest, dass Janet den Fehler nicht alleine beheben kann und die Hilfe von einem Fachmann benötigt. Dazu muss ein Reparaturtermin vereinbart werden. Das System zeigt Janet den Menüpunkt *Reparaturtermin vereinbaren* an.

7. Janet hat noch Garantie auf das Gerät und aktiviert den Menüpunkt *Reparaturtermin vereinbaren*. Janets Terminkalenderanwendung verhandelt mögliche Termine für die Reparatur und schlägt ihr diese vor. Janet kann sich für einen Termin entscheiden. Dieser wird auch gleich in ihren elektronischen Kalender eingetragen und damit ist das Problem vorerst für sie gelöst (automatische Terminvereinbarung).

Szenario 2

Szenario: Janet sitzt vor dem Fernseher und das Telefon klingelt

Akteur-Instanzen:

- Janet: Bewohnerin
- Anrufer
- Telefon, Fernseher: Geräte im Haushalt

Ereignissfluss:

1. Janet schaut sich gerade eine interessante Reisereportage im Fernsehen, an als das Telefon klingelt.
2. Auf dem Fernseher erscheint ein Fenster mit dem Namen des Anrufers und der Frage, ob Janet das Telefonat annehmen und die Sendung aufzeichnen möchte. Über ein einfaches Menü (Drücke Grüne Taste für Annehmen, Drücke Rote Taste für Besetzt) kann die Auswahl getroffen werden.
3. Janet möchte den Anruf annehmen. Von diesem Zeitpunkt an wird das aktuelle Programm angehalten und auf einem beliebig verfügbaren Videoaufnahmegerät (z.B. dem angeschlossenen Festplattenrekorder) aufgezeichnet.
4. Das Telefonat wird über die Fernseherlautsprecher und ein Raummikrofon geführt. Janet, die nicht mehr gut hört und auch Probleme hat, die kleinen Tasten auf dem Telefon zu bedienen, kann so ganz bequem von der Couch ihre Telefonate führen. Ein weiterer Vorteil ist, dass sie ihr Telefon nicht mehr suchen muss und dass die Anrufer nicht mehr den, meistens sehr lauten Fernseher, im Hintergrund hören.
5. Nach dem Telefonat wird ihr auf dem Fernseher ein Auswahlmenü angezeigt, auf dem angeboten wird die Reisereportage fortzusetzen.
6. Janet drückt die OK-Taste. Die Wiedergabe der Reportage wird ab diesem Zeitpunkt fortgesetzt.

Szenario 3

Szenario: Janet befindet sich in der Küche, und ein Nachbar klingelt an der Wohnungstür

Akteur-Instanzen:

- Janet: Bewohnerin
- Nachbar
- Telefon, Türwechselsprechanlage mit integrierter Türüberwachungskamera, Mikrowelle: Geräte im Haushalt

Ereignissfluss:

1. Janet befindet sich gerade in der Küche, als ein Nachbar an der Wohnungstür klingelt. Die Türwechselsprechanlage ist mit einer Videokamera ausgestattet.
2. Das System ermittelt den Aufenthaltsort von Janet und sucht nach einem Gerät, welches zur Anzeige dieses Ereignisses geeignet ist. In der Küche befindet sich ein

Telefon. Das System lässt das Telefon klingeln, da dieses so eingestellt ist, dass es klingeln soll, wenn jemand an der Tür klingelt.

3. Auf dem Telefondisplay wird der Text Türwechselsprechanlage angezeigt. Janet nimmt den Hörer ab und spricht mit dem Nachbarn vor der Tür.

4. Sie dreht sich dann zu ihrer Mikrowelle um, die einen eingebauten Bildschirm (Videoausgabegerät) hat und sieht dort das eingeblendete Videobild von der Türüberwachungskamera.

5. Auf dem Display der Mikrowelle wird ein Menü angezeigt mit möglichen Aktionen, die in der aktuellen Situation von Janet ausgeführt werden können. Die Mikrowelle hat Tasten, mit denen Einstellungen vorgenommen und Menüpunkte ausgewählt werden können.

6. Janet wählt auf der Mikrowelle die Aktion *Wohnungstür öffnen* aus.

7. Die Wohnungstür wird geöffnet, und der Nachbar betritt die Wohnung.

Die Szenarien beschreiben drei verschiedene Anwendungsbereiche. Das erste Szenario steht stellvertretend für die Anwendungsfälle, in denen ein Gerät einen fehlerhaften Zustand hat. Auf einem Gerät tritt ein Fehler auf, und das Gerät löst daraufhin ein Ereignis aus. Diese Ereignisse können entweder direkt auf dem Gerät angezeigt oder werden auf einem Gerät dargestellt werden, welches sich der Wahrnehmung eines Nutzers (beziehungsweise dem, den das Ereignis betrifft) befindet. Der Nutzer soll auf das Ereignis reagieren. Dazu muss das Ereignis auf einem Gerät dargestellt werden, auf dem der Nutzer auch für ein anderes Gerät Interaktionen ausführen kann (z.B. TV). So kann der Nutzer mit seinen Interaktionen, wahlweise von verschiedenen Geräten aus, auf ein entferntes Ereignis reagieren.

Das zweite Szenario beschreibt eine Situation, in der ein Ereignis auftritt und vom System Regeln ausgeführt werden um dem Nutzer mögliche Aktionen vorzuschlagen und diese automatisch auszuführen, wenn das den Präferenzen des Nutzers entspricht.

Das dritte Szenario beschreibt den Anwendungsbereich der Fernbedienungsfunktion (Geräte werden aus der Ferne bedient). Dazu werden auf einem Gerät die Steuerungsfunktionen dargestellt, welche benötigt werden um ein anderes Gerät zu bedienen (Beispiel: Öffnen der Wohnungstür über ein Menü auf der Mikrowelle).

2.2 Anwendungsfälle

Aus den Szenarien wurden Anwendungsfälle identifiziert, welche näher betrachtet und von der zu entwickelnden Middleware-Lösung umgesetzt werden sollen. Die Abbildung 2.1 zeigt die Kommunikationsbeziehungen zwischen den Akteuren *Nutzer* und *System* und den Anwendungsfällen, die erfasst werden sollen. In diesem Diagramm werden abstrakte Anwendungsfälle dargestellt, die als Oberklassen der konkreten Anwendungsfälle dienen. So müssen nicht alle Anwendungsfälle, die in einem Smarthome auftreten können, erläutert werden.

Auf der linken Seite ist der Akteur *Nutzer* dargestellt, der mit den verschiedenen An-

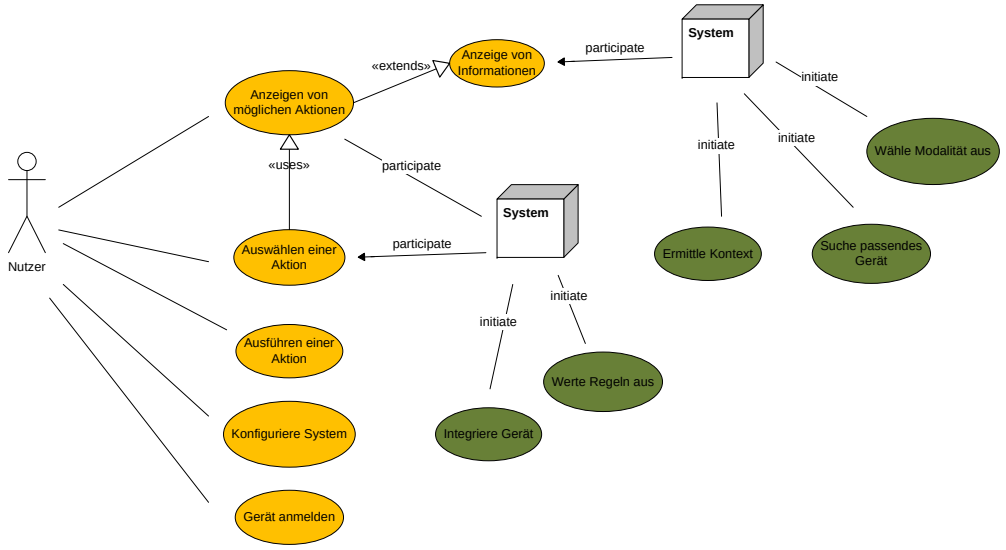

Abbildung 2.1: Kommunikationsbeziehungen zwischen Anwendungsfalldiagrammen

wendungsfällen interagiert. Auch das *System* ist als Akteur dargestellt, der an den Anwendungsfällen partizipiert und für deren Umsetzung weitere Anwendungsfälle initialisiert und ausgeführt werden.

Wird ein Ereignis im System ausgelöst, dann wird der Anwendungsfall *Anzeige von Informationen* gestartet. Dem Nutzer werden die Informationen auf einem beliebigen Gerät dargestellt. Der Anwendungsfall *Anzeigen von Aktionen* ist als Erweiterung von *Anzeige von Informationen* modelliert. Aktionen sind Informationen, auf die der Nutzer reagieren kann. Damit Informationen dargestellt werden können, muss das System die Modalität auswählen, mit der die Informationen dargestellt werden. Es muss nach einem passenden Gerät suchen und den Kontext ermitteln. Das System initialisiert dazu die entsprechenden Anwendungsfälle. Der Nutzer führt den Anwendungsfall *Auswählen einer Aktion* aus. Das System partizipiert an diesem Anwendungsfall, indem es die beiden Anwendungsfälle *Werte Regeln aus* und *Integriere Gerät* initialisiert. Das System ermittelt anhand von Regeln und der integrierten Geräte im System, welche Aktionen dargestellt und welche Aktionen vom Nutzer ausgewählt werden. Der Anwendungsfall *Konfiguriere System* kennzeichnet die Situationen, in denen der Nutzer Änderungen am System vornimmt. Dazu gehört das Erstellen, Modifizieren und Löschen von Regeln und Policen. Den Anwendungsfall *Gerät anmelden* initialisiert der Nutzer wenn er ein neues Gerät anmelden möchte. Der Nutzer registriert Geräte und meldet diese durch Auswahl dieser Funktion in einer Auswahlmaske an. Alternativ meldet das System Geräte automatisch an, wenn sich diese in der Nähe oder im Besitz des Nutzers befinden, wenn dieser das Haus betritt.

Es wurden verschiedene Akteure in diesen Anwendungsfällen identifiziert. Ein Akteur kann ein Nutzer (Mensch), aber auch ein anderes System sein, das dieses System nutzt. Der Akteur befindet sich außerhalb der Systemgrenze, ist also extern. Geräte sind einerseits Teil des Systems aber auch Akteure. Den Akteuren können folgende Rollen zugeordnet werden.

- Nutzer
- Bewohner

- Freund des Bewohners
- Familienangehöriger
- Schornsteinfeger
- Nachbar
- Anrufer
- System
- Gerätehersteller
- externe Terminplanungsanwendung eines Geräteherstellers
- Systemadministrator
- Gerät

Für das bessere Verständnis der Anwendungsdomäne werden für die oben genannten abstrakten Anwendungsfälle die folgenden detaillierten Beispiele aufgeführt.

Anwendungsfall 1

Anwendungsfallname *Akteure*	Anzeigen des Gerätestatus Initiiert von Bewohner Kommuniziert mit abzufragendem Gerät
Ereignisfluss	1. Der Bewohner ruft das Auswahlmenü zur Abfrage eines Gerätestatus auf. 2. System zeigt vorhandene Geräte an. 3. Der Bewohner wählt ein Gerät aus der Liste aus. 4. Das System fragt den Status des Gerätes ab. 5. Das Gerät schickt seinen Status an das System. 6. Das System leitet die Statusmeldung an das Gerät weiter, das die Aufmerksamkeit des Bewohners hat.
Anfangsbedingungen *Abschlussbedingungen* *Qualitätsanforderungen*	Keine Statusmeldung wurde auf dem Display angezeigt. Der Bewohner erhält die gewünschte Statusmeldung innerhalb von 5 Sekunden.

Anwendungsfall 2

Anwendungsfallname	Steuere Gerät fern
Akteure	Initiiert von Bewohner Kommuniziert mit fernzusteuerndem Gerät
Ereignisfluss	1. Der Bewohner ruft das Auswahlmenü zur Fernsteuerung eines Gerätes auf. 2. System zeigt vorhandene Geräte an. 3. Der Bewohner wählt ein Gerät aus der Liste aus. 4. Das System ruft die Kontrollfunktionen des Gerätes ab. 5. Das Gerät schickt seine Kontrollfunktionen an das System. 6. Das System leitet die Kontrollfunktionen an das Gerät weiter, das die Aufmerksamkeit des Bewohners hat. 7. Dieses Gerät zeigt die Kontrollfunktionen an. 8. Der Bewohner wählt eine Kontrollfunktion aus und stellt Parameter ein. 9. Das System übermittelt diese Kontrollfunktionen und die Parameter an das Gerät. 10. Das Gerät führt die Kontrollfunktionen aus. 11. Das System zeigt dem Nutzer eine Rückmeldung über die ausgeführte Aktion an.
Anfangsbedingungen	Es muss ein Gerät vorhanden sein, welches ferngesteuert werden kann.
Abschlussbedingungen	Der Bewohner hat eine Bestätigung über die Ausführung einer Funktion erhalten.
Qualitätsanforderungen	Der Bewohner erhält die gewünschte Bestätigung innerhalb von 5 Sekunden.

Anwendungsfall 3

Anwendungsfallname	Präsentation von Informationen
Akteure	Bewohner Anrufer Gerät
Ereignisfluss	1. Der Anrufer ruft den Bewohner auf einem Gerät an. 2. Das Gerät meldet das Ereignis an das System. 3. Das System sucht ein Gerät, das die Aufmerksamkeit des Bewohners hat und welches dieses Ereignis anzeigen kann. 4. Das Gerät zeigt dem Bewohner das Ereignis und die Aktionen, um darauf zu reagieren, in einem Auswahlmenü an. 5. Der Bewohner wählt eine Aktion aus dem Auswahlmenü aus. 6. Das Gerät schickt die ausgewählte Aktion an das System. 7. Das System führt diese Aktion aus. 8. Der Bewohner führt ein Telefonat.
Anfangsbedingungen	Es muss ein Gerät vorhanden sein, welches angerufen werden kann. Der Bewohner muss erreichbar sein.
Abschlussbedingungen	Der Bewohner wurde über das Ereignis informiert und es wurde eine Auswahl an Aktionen angezeigt.
Qualitätsanforderungen	Dem Bewohner wird die Information über das Ereignis innerhalb von 1 Sekunde angezeigt.

Anwendungsfall 4

Anwendungsfallname	Nachbar klingelt an Wohnungstür
Akteure	Bewohner Nachbar Gerät
Ereignisfluss	1. Nachbar klingelt an der Wohnungstür. Die Wechselsprechanlage löst ein Ereignis aus und sendet dieses an das System. Wenn es klingelt soll der Bewohner darüber informiert werden. 2. Das System ermittelt den Standort des Bewohners und überprüft, ob sich dort Geräte befinden, die das Ereignis ausgeben können. 3. Die Bewohnerin befindet sich in der Nähe der Mikrowelle. 4. Das System schickt das Ereignis an die Mikrowelle. Auf der Mikrowelle öffnet sich ein Anzeigefenster. 5. Die Bewohnerin wählt aus, dass sie mit der Person vor der Tür sprechen möchte. 6. Die Middleware baut eine Gesprächsverbindung zwischen der Mikrowelle und der Türwechselsprechanlage auf. 7. Auf der Mikrowelle werden mögliche Aktionen angezeigt, die in dieser Situation ausgeführt werden können. 8. Die Bewohnerin möchte den Nachbarn einlassen und wählt die Aktion *Öffne Wohnungstür* aus (siehe Anwendungsfall *Steuere Gerät fern*). 9. Das System öffnet die Wohnungstür und der Nachbar kann die Wohnung betreten. 10. Das Anzeigefenster auf der Mikrowelle wird geschlossen.
Anfangsbedingungen	Es muss ein Gerät vorhanden sein, auf dem der Bewohner informiert werden kann. Der Bewohner muss erreichbar sein.
Abschlussbedingungen	Der Nutzer wurde über das Ereignis informiert und es wurde eine Auswahl an Aktionen angezeigt.
Qualitätsanforderungen	Dem Nutzer wird die Information über das Ereignis innerhalb von 1 Sekunde angezeigt.

Anwendungsfall 5

Anwendungsfallname	Löse Aktion aus
Akteure	Bewohner Gerät
Ereignisfluss	1. Ereignis tritt auf. 2. Das System zeigt auf einem Gerät mögliche Aktionen an *Anzeigen von möglichen Aktionen.* 3. Der Bewohner wählt eine Aktion aus *Auswählen einer Aktion.* 4. Der Bewohner startet die Ausführung der Aktion *Ausführen einer Aktion.* 5. Das System führt die Aktion aus. 6. Das System informiert den Bewohner über die korrekte Ausführung der Aktion oder über Ausführung mit Fehler.
Anfangsbedingungen	Es muss ein Gerät vorhanden sein, auf dem der Bewohner informiert werden kann. Der Bewohner muss erreichbar sein.
Abschlussbedingungen	Der Bewohner wurde über das Ereignis informiert und es wurde eine Auswahl an Aktionen angezeigt, aus denen der Bwohner eine ausgewählt und ausgeführt hat.
Qualitätsanforderungen	Dem Bewohner wird die Information über das Ereignis innerhalb von 1 Sekunde angezeigt.

2.3 Anforderungsanalyse

Während der Anforderungsanalyse wurden aus den dargestellten Szenarien und mit Hilfe der daraus entwickelten Anwendungsfälle Anforderungen ermittelt. Das sind erstens Anforderungen, die ein Nutzer an ein intelligentes System hat. Zweitens sind es Anforderungen, die an die beteiligten Geräte gestellt werden. Eine Unterteilung nach Muss- und Kann-Kriterien wird im Abschluss dieses Abschnitts vorgenommen.

2.3.1 Funktionale Anforderungen

Aus der Anforderungsanalyse haben sich diese funktionalen Anforderungen an das zu entwickelnde System ergeben.

I/ **Integration heterogener Geräte (PC, Haushaltselektronik,**

Unterhaltungselektronik, eingebettete Geräte,mobile Geräte)

Heterogene Geräte sollen sich dynamisch anmelden und abmelden können (Service Discovery und Integration). Die Gerätekategorien umfassen PCs, Unterhaltungselektronikgeräte, Haushaltsgeräte und Mobile Geräte. Sollten in der Zukunft neue Geräteklassen hinzukommen, dann sollen diese Geräte auch integrierbar sein. Das System soll herstellerunabhängig sein. Das System soll unabhängig von Netzwerkschnittstellen und dem Netzwerkprotokoll sein. Das Gerät soll unabhängig von dem verwendeten Betriebssystem integriert werden können. Die Kooperation zwischen verfügbaren heterogenen Geräten soll ermöglicht werden. Teilnehmende Geräte sollen dem System ihre Dienste (Funktionalität) zur Verfügung stellen. Wie die Kommunikation zwischen Geräten erfolgt, spielt keine Rolle. Entscheidend ist die semantische Integration, unabhängig von einem bestimmten Protokoll. Der Nutzer soll Geräte anmelden können.

II/ **Zwischen Geräten soll eine nahtlose Kooperation, ohne vorherige Konfiguration, möglich sein.**

Der Nutzer soll dem Geräte nicht mitteilen müssen, wie es mit einem anderen Gerät kommuniziert oder welche Funktionalitäten es von diesem Gerät anzeigt. Kooperation von beliebigen Geräten, um dem Nutzer die beste Bedienung des Systems zu ermöglichen.

Beispiel: Das aktuelle Fernsehprogramm auf einem beliebigen Aufnahmegerät aufnehmen.

III/ **Geräte müssen eine semantische Selbstbeschreibung haben.**

Geräte beschreiben ihre Funktionalität semantisch, so dass diese vom System verstanden werden können. Die Geräte sollen sich selbst beschreiben, so dass ihre Funktionalitäten vom System verstanden werden und in das Gesamtsystem integriert werden können.

IV/ **Finden von Geräten mit Suche nach deren Fähigkeiten.**

Das System soll Geräte finden können, durch die Suche nach semantischen Beschreibungen der Geräteeigenschaften und Gerätefunktionalitäten.

V/ **Das System muss ein geeignetes Gerät für die Interaktion mit dem Nutzer ermitteln können.**

VI/ **Es soll ein Repository für Geräte geben.**

Darin werden alle Geräte verwaltet, die sich in der Hausdomäne befinden und dem Nutzer gehören. Fremde Geräte müssen freigeschalten werden bevor diese sich anmelden können.

VII/ **Das System soll kontextbewusst sein und sich zu jeder Zeit an den Kontext des**

Nutzers (Aufenthaltsort, Aktivität) und seine Nutzerpräferenzen anpassen.

Das System muss den Kontext, der für das kontextabhängige Systemverhalten benötigt wird, ermitteln und darauf zu jedem Zeitpunkt Zugriff haben. Kontextänderungen müssen registriert werden. Das System soll die aktuelle Situation des Nutzers erfassen und erkennen (Context-awareness). Fremde Nutzer, die in das Haus kommen, sollen vorher gefragt werden, ob sie an dem System teilnehmen möchten. Das System soll aus dem Nutzerverhalten in Situationen lernen und sich anpassen können. Der Nutzer kann seine Präferenzen einstellen, dass heißt, wie sich das System in einer bestimmten Situation verhalten soll. Das System muss die Hausdomäne semantisch verstehen. So muss das System dem Bewohner bei schlechtem Wetter empfehlen, die Fenster zu schließen oder diese automatisch schließen, wenn eine Gerätesteuerung vorhanden ist. Das System überwacht, bei Vorhandensein entsprechender Geräte, den Gesundheitszustand des Nutzers und löst einen Alarm aus, wenn vorgegebene Parameter überschritten werden.

VIII/ **Anzeige von Informationen**

Auf einem geeigneten Gerät, welches sich in der Nähe des Nutzers befindet, sollen beliebige Informationen (wie Warnung, Fehler, Status, Notfall, Termin) und Ereignisse anderer Geräte anzeigt werden. Das System muss ermitteln, wo sich der Nutzer aufhält und vom welchem Gerät er das Ereignis wahrnimmt. Dann muss das System die Darstellung der Information an dieses Gerät anpassen. Die Informationen sollen in einer für ältere Menschen verständlichen Weise dargestellt werden. Jedes Ereignis soll ausführlich mit einer Beschreibung dargestellt werden. Dazu gehört die Information, von welchem Gerät aus, das Ereignis ausgelöst worden ist.

- Das System muss die Priorität einer Nachricht kennen. Konflikte in der Behandlung von Ereignissen dürfen nicht auftreten. Wenn zur gleichen Zeit eine Warnsituation und ein Notfall auftreten, dann muss der Notfall zuerst angezeigt werden, da diese Nachricht eine höhere Priorität hat.
- Bei einem Notfall muss der Nutzer oder eine andere Person, die Hilfe einleiten kann, auf jeden Fall informiert werden. Dem Nutzer muss eine Handlungsvorschlag dargestellt oder ausgegeben werden.
- Für die vollständige Darstellung einer Warnung muss dem Nutzer mitgeteilt werden, was er tun muss.
- Ebenfalls wichtig ist die Überwachung von Gerätefunktionen, sowie die Statusabfrage aller im System vorhandenen Geräte.
- Es müssen Kontrollfunktionen auf einem geeigneten Gerät angezeigt werden können.
- Auf einem geeigneten Gerät muss eine Übersicht aller verfügbaren Geräte und deren Dienste abrufbar sein.
- Bei Ereignissen, die eine Reaktion erfordern (Notfall) soll jemand informiert werden, erst Recht wenn der Nutzer nicht erreichbar ist oder nicht auf das Ereignis reagiert. Es muss dazu eine Prioritätsliste geben, in der die Reihenfolge

der Benachrichtigungen festgelegt wird: Familie, Nachbar, Feuerwehr (Weiterleitung von Ereignissen). Der Nutzer muss diese Ereigniskette definieren können.

- Ereignisse werden dem Nutzer oder einer vorher definierten Gruppe von Nutzern angezeigt. Dem Nutzer werden nur Ereignisse angezeigt, die ihn interessieren, oder die er registriert hat (Einstellen von Nutzerpräferenzen).

Beispiel: Wenn jemand an der Tür klingelt, soll der Nutzer darüber auf einem Gerät informiert werden, mit dem er gerade interagiert. Oder der Nutzer soll über Informationen, auf einem Gerät, benachrichtigt werden, welches er als erstes abruft (falls er nicht daheim war). Wenn der Nutzer nicht da ist, wird sein Nachbar darüber informiert, dass die Waschmaschine des Nutzers ausläuft.

IX/ **Anzeige von möglichen Aktionen, die mit Geräten ausgeführt werden und Ausführen einer Aktion**

Dem Nutzer sollen mögliche Aktionen dargestellt werden, abhängig davon wo er sich aufhält und mit welchen Geräten er in Kommunikation steht. Diese können in seiner momentanen Situation auf den verfügbaren Geräten ausgeführt werden. Zu einem Ereignis, welches eine Reaktion erfordert, soll dem Nutzer eine Auswahl aller möglichen Aktionen angezeigt werden, um auf das dargestellte Ereignis reagieren zu können. Das System muss dazu ein Gerät ermitteln, welches die Möglichkeit zur Interaktion bietet, so dass der Nutzer Aktionen auswählen und ausführen kann.

- Bei einer Fehlermeldung werden dem Nutzer Aktionen angezeigt, so dass er unter Anleitung des Systems den Fehler beheben kann.
- Der Nutzer soll einen Anruf auf einem beliebigen Gerät annehmen können.
- Anzeige von Ereignissen, die Interaktion erfordern (Ereignis anzeigen, oder zum Beispiel den Händler informieren)

Beispiel: Wenn jemand an der Tür klingelt soll der Nutzer darüber auf einem Gerät informiert werden. Auf einem Gerät sollen die möglichen Aktionen für dieses Ereignis angezeigt werden. Auf dem Gerät wird der Name der klingelnden Person angezeigt und die Aktionen „Tür öffnen“, „Mit der Person sprechen“ oder „Ablehnen“. Der Nutzer wählt die Aktion „Tür öffnen“ und die Tür wird vom System geöffnet.

X/ **Kontrollieren von Funktionen eines anderen Gerätes von einem geeigneten Gerät aus (Fernbedienung)**

Geräte, die sich im Haus befinden, sollen sich über ein beliebiges interaktionsfähiges Gerät fernbedienen lassen.Auf diesem Gerät soll ein Auswahlmenü zur Auswahl eines Gerätes dargestellt werden. Von dem ausgewählten Gerät sollen alle Funktionen, die aus der Ferne aufgerufen werden können, dargestellt werden. Aus diesen Funktionen wählt sich der Nutzer eine Funktion oder mehrere Funktionen aus, die er ausführen möchte.

Beispiel: Der Nutzer schaltet die Waschmaschine vom Fernseher aus.

XI/ **Das System muss die geeignete Modalität zur Ausgabe und Eingabe von Informationen auswählen.**

Die Eingaben und Ausgaben in das System sollen unabhängig von der Modalität erfolgen. Das System wählt für die Anzeige von Informationen die geeignete Modalität aus. Das ist abhängig von der Situation, den verfügbaren Geräten und den Nutzerpräferenzen. Folgende Eingabemodalitäten sollen unterstützt werden: Per Tastatur, Maus, Stift, Sprache, Gesten, Mimik. Folgende Ausgabemodalitäten sollen unterstützt werden: visuell, akustisch, Gestik, Braille.

XII/ **Nutzer soll das System konfigurieren können.**

Der Nutzer soll zu jedem Zeitpunkt das System konfigurieren können. Das System muss es dem Nutzer ermöglichen, anwendungsspezifische Regeln und Policen zu definieren und auszuführen. Dazu muss es Regeln verwalten und speichern können. Regeln müssen sich vom Nutzer erstellen lassen. Die Regeln sollen so beschaffen sein, dass der Nutzer aus vorgefertigten Komponenten die Regeln zusammensetzen kann. Diese Komponenten müssen leicht verständlich sein. Wenn Regeln für Geräte beschrieben werden, dann sollen dem Nutzer das Gerät und die Gerätefunktionen in einer Auswahl dargestellt werden. Die Situation in der sich der Nutzer befindet muss erfasst werden können.

XIII/ **Das System soll es ermöglichen, dass ein Gerät einen Fehlerbericht an den Händler sendet**

Das System soll als Antwort einen Lösungsvorschlag erhalten. Wenn der Nutzer das Problem nicht beheben kann, soll das System mit dem Händler automatisch einen Termin mit dem Gerätehersteller vereinbaren können.

XIV/ **Dem Nutzer sollen nur Aufgaben angezeigt werden, die seiner Intention und seinem Kontext entsprechen.**

Es muss ermittelt werden können, was die Intention des Nutzers ist. Dazu muss analysiert werden, wie sich der Nutzer in bisherigen Situationen entschieden hat und diese Entscheidung in einer ähnlichen Situation zuerst vorgeschlagen werden. Bei der Kooperation von Geräten gibt es unterschiedlich wahrscheinliche Aufgaben, die der Nutzer ausführen möchte. Diese sollen, nach der Wahrscheinlichkeit, sortiert angezeigt werden.

XV/ **Das System soll anhand von Regeln automatisch Entscheidungen treffen können.**

Dazu muss das System Regeln verwalten und ausführen können. Dem Nutzer werden Aufgaben abgenommen, wenn das System Geräte automatisch für den Nutzer konfiguriert. Das System hilft dem Nutzer bei der Überwachung und Kontrolle von Geräten im Haushalt. Der Nutzer soll Regeln für bestimmte Situationen erstellen können.

XVI/ **Das System soll den Nutzer an Tätigkeiten und Termine erinnern können.**

Der Nutzer kann Termine auf Aufgaben in das System eingeben und das System kann automatisch Aufgaben ermitteln, aus einem geplanten Sollzustand und dem aktuellen Zustand. Der Nutzer soll an Geburtstage, Treffen mit Freunden und der Familie, an die Tabletteneinnahme erinnert werden. Das System soll Termine mit einer Terminplanungsanwendung abgleichen können. Der Nutzer soll einen Sollzustand für jedes Gerät eingeben können.

Beispiel: Das System kennt den Mindestinhalt des Kühlschrankes und erstellt automatisch eine Einkaufsliste für fehlende Produkte oder bestellt die Produkte bei einem Internethändler, je nach Nutzerwunsch. Der Nutzer und sein Verhalten in den jeweiligen Situationen wird beobachtet und die Regeln werden angepasst.

XVII/ **Weiterführung der Bedienung auf einem anderen Gerät (Task Continuation).**

Der Nutzer soll die Anzeige vom einem Gerät auf ein anderes Gerät weiterleiten können (Weiterleitung). Das Auswahlmenü wandert von Gerät zu Gerät. Auf dem anderen Gerät wird der gleiche Inhalt dargestellt. Dazu gehört eine Auswahl eines Gerätes, Parameter die schon eingeben wurden, werden übergeben. Dazu muss das Gerät seinen aktuellen Zustand an das System übermitteln können. Das System findet das neue Gerät, welches der Nutzer bedienen möchte und stellt dort den Zustand des vorherigen Gerätes her. Transferieren des Dienstes von einem Gerät zu einem Anderen soll ermöglicht werden (Dienstemigration).

XVIII/ **Der Nutzer bzw. das System müssen jederzeit einen Alarm absetzen und Hilfe anfordern können.**

Mit einem Knopfdruck muss Alarm ausgelöst werden können. Das System soll Alarmsituationen selbständig erkennen und für den Nutzer Hilfe anfordern. Dieser Fall trifft unter anderem zu, wenn eine Fehleranzeige erfolgt ist aber der Nutzer nicht reagiert. Die Wahrscheinlichkeit eines Fehlalarms muss unter zwei Prozent sein.

Löse Notfall aus mit HealthMonitoringService. Der Gesundheitszustand des Nutzers soll die gesamte Zeit überwacht werden. Wenn die Überwachung für mehr als 20 Sekunden ausfällt, dann muss sofort ein Alarm ausgelöst werden. Das Alarmsystem muss konfigurierbar sein. Der Nutzer kann einstellen, wer alarmiert werden soll.

XIX/ **Protokollierung aller Nutzerinteraktionen mit den Geräten.**

Es muss erfasst werden, welche Eingabe der Nutzer gemacht hat und wie sich das Systemverhalten dadurch verändert hat. Es soll protokolliert werden wer, mit einem Gerät interagiert hat.

Beispiel: Janet hat die Waschmaschine angestellt und ein Waschprogramm gestartet. Daher wird angenommen, dass sie sich für die Ereignisse interessiert, die während des Waschprogramms auftreten.

2.3.2 Nichtfunktionale Anforderungen

Mit den Nichtfunktionalen Anforderungen werden die Anforderungen an das System beschrieben, die nicht direkt die Systemfunktionalität betreffen.

1. Das System muss es dem Nutzer ermöglichen, dass sein Fokus bei der Bedienung der Geräte ist: *„Was möchte ich tun?“* und nicht *„Wie muss ich es tun?“*. Der Nutzer soll seine Aufmerksamkeit auf seine wesentlichen Aufgaben richten können.
2. Das System soll Regeln so ausführen, dass der Nutzer keine Verzögerung bemerkt. Das heißt die Verzögerung darf nicht länger sein, als die Ausführung einer bestimmten Aktion normalerweise in dieser Anwendungsdomäne beträgt.
3. Zeitnahe und garantierte Anzeige von Ereignissen.
4. Leistung und Effizienz (Antwortzeiten, Ressourcenbedarf)
5. Sicherheitsanforderungen werden nicht betrachtet, da zu umfangreich. Untersucht werden müssten Aspekte wie Vertraulichkeit, Datenintegrität, Verfügbarkeit.
6. Zuverlässigkeit (Robustheit, Fehleranfälligkeit Schutz vor Fehlbedienung, Systemverhalten muss immer deterministisch sein)
7. Usability bzw. Anforderungen an die Benutzerschnittstelle

    - Es soll die Absicht des Nutzers aus dem Kontext ermittelt werden und es sollen nur die Tasks angezeigt werden, welche ausgeführt werden können.
    - Der Nutzer soll nachvollziehen können, welche Regeln das System ausgeführt hat.
    - Der Nutzer soll in die Ausführung von Regeln jederzeit eingreifen können.
    - Das System soll eine uniforme (homogene) Bedienung und Systemverhalten durch einheitliche Benutzerschnittstellen bieten. Das gilt auch unterschiedliche Geräte und Geräte gleicher Kategorie aber von unterschiedlichen Herstellern.
    - Die Kooperation der Geräte soll dahingehend optimiert werden, dass dem Nutzer die beste Usability geboten wird.
    - Verbesserung der Usability von Geräten. In jeder Situation soll das Gerät mit der besten Nutzbarkeit für den Nutzer ausgewählt werden (Darstellen einer Textnachricht wie SMS auf dem Fernseher, da dieser eine größere Schrift darstellen kann).
    - Ältere Menschen sollen das System bedienen können. Zu dieser Thematik müssen Usability-Tests durchgeführt werden (Prototypisches Haus bauen).
    - Die Benutzerschnittstelle muss sich an den Kontext des Nutzers anpassen, dass heißt an seinen Ort, das Interaktionsgerät oder seine Behinderungen.
    - Barrierefreie Bedienung muss gegeben sein. Interaktion mit Geräten muss mit eingeschränktem Seh- und Hörvermögen möglich sein. Beispielsweise Verwendung von kontrastreicher Schrift. Keine Verwendung von hohen Tönen. Die

Geräte/intelligenten Gegenstände sollen weitestgehend unmerklich den Nutzer unterstützen.

8. Technologische Erweiterbarkeit, das heißt das System muss neue Geräte und Technologien integrieren können. Das System muss aufwärts- als auch abwärtskompatibel sein, um Inkompatibilitäten zwischen verschiedenen Versionen zu reduzieren.

9. Das System soll sich leicht installieren lassen.

10. Wartbarkeit und Wartungsfreundlichkeit

2.4 Zusammenfassung

Dieses Kapitel hat einen kurzen Einblick in die Anwendungsdomäne gegeben, in deren Umfeld diese Diplomarbeit angesiedelt ist. Es wurden am Anfang Szenarien vorgestellt, die den Einsatz dieses Systems in einem intelligenten Haus, vom Gesichtspunkt eines Nutzers aus, verdeutlichen sollten. Aus den Szenarien wurden Anwendungsfälle ermittelt, die die gewünschte Funktionalität des Systems beschreiben. Mit Hilfe der Anwendungsfälle wurden dann Funktionale und Nichtfunktionale Anforderungen ermittelt. Diese Anforderungen können nur eine Teilmenge der Anforderungen sein, die an ein System gestellt werden, welches die Intelligenz in einem Smarthome realisieren soll. Die Nichtfunktionalen Anforderungen sollen ebenfalls nur einen Eindruck vermitteln für die Vielzahl an verschiedenen Aspekte, die in einem solchen System bis zur Markteinführung berücksichtigt werden müssten. Aspekte wie Usability, Sicherheit und Zuverlässigkeit sind die größten Herausforderungen, die gelöst werden müssen bevor ein reales System von den Nutzern akzeptiert und schlussendlich eingesetzt wird.

In den folgenden Kapiteln dieser Diplomarbeit werden zunächst verwandte Arbeiten und Technologien in diesem Bereich vorgestellt. Dann folgen die Kapitel zur Konzeption, Vorstellung des Prototyps, Verifikation der Konzeption und eine abschließende Zusammenfassung.

3 Verwandte Arbeiten und existierende Technologien

Diese Kapitel beschreibt existierende Technologien und verwandte Arbeiten, die an dieser Stelle vorgestellt werden. Im folgenden Abschnitt werden Begriffe Ontologien und Regeln eingeführt. Im zweiten Teil wird auf verwandte Forschungsarbeiten im Bereich des Ubiquitous Computing, Smarthomes und kontextbewussten Infrastrukturen eingegangen.

3.1 Ontologien und Regeln

3.1.1 OWL

Die Web Ontology Language (OWL) [19] wird für die Beschreibung von Ontologien verwendet. Eine Ontologie dient der Beschreibung von Begriffen und Relationen zwischen diesen. Ein häufig zitiertes Zitat von Tom Gruber aus dem Jahr 1993.

> An ontology is a formal explicit description of concepts and relations in a domain of discourse, including properties of each concept and constraints expressed as axioms [10].
>
> Eine Ontologie ist eine formale und explizite Beschreibung von Konzepten und Beziehungen in einer Anwendungsdomäne und beinhaltet Attribute jeden Konzeptes und Einschränkungen, die als Axiome dargestellt werden [10].

Ontologien werden nach dem Level der Allgemeinheit (Generality) klassifiziert in Top-Level-Ontologien, in Domänen- und Task-Ontologien und in Anwendungsontologien.

Die Hauptcharakteristiken von Ontologien sind:

- Teilen des allgemeinen Verständnisses einer Informationsstruktur zwischen Menschen und Softwareagenten.
- Ermöglichen die Wiederverwendung von Domänenwissen.
- Ermöglichen Annahmen von expliziten Domänen.
- Trennung des Domänenwissens von dem operationellen Wissen.

Durch die Nutzung von Ontologien ergeben sich die folgenden Vorteile: Intelligenteres Verhalten, Flexibilität, Wiederverwendbarkeit und semantische Interoperabilität. Mit Ontologien ist auch die Beschreibung von Webservices möglich, Agenten entdecken diese automatisch und greifen darauf zu. Es können damit die Schnittstellen, die Parameter und die Funktionalität beschrieben werden.

3.1.2 Semantic Web Rule Language (SWRL)

Die Semantic Web Rule Language (SWLR) [14] ist ein Vorschlag für eine Regelsprache zur Beschreibung des Semantic Web welche die Untersprachen von OWL (OWL DL und OWL Lite) kombiniert mit denen der Rule Markup Sprache (RuleML).

Die Regeln entsprechen der Form Vorbedingung(antecedent) $\longrightarrow$ Konsequenz (consequent)) Wenn alle Vorbedingungen erfüllt sind dann ergibt sich daraus die Konsequenz. Eine Beispielregel in menschenlesbarer Syntax lautet wie folgt: hatEltern(?x,?y) $\wedge$ hatBruder(?y,?z) $\Rightarrow$ hatOnkel(?x, ?z).

3.1.3 Context-Awareness

Eine der am häufigsten zitierten Definitionen für Kontext stammt von Anind K. Dey.

> Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [1].
>
> Kontext ist jede Information, die genutzt werden kann, um die Situation einer Entität zu beschreiben. Eine Entität kann eine Person, Platz oder Objekt sein, welches als wichtig erachtet wird für die Interaktion zwischen einem Nutzer und einer Anwendung, den Nutzer und die Anwendung einschließend [1].

Dass heißt unter dem Begriff Kontext werden sämtliche Informationen verstanden, die Umstände und Ausprägungen einer Situation charakterisieren und relevant für Interaktionen zwischen Anwendern und einer Anwendung sein können. Dabei sind der Anwender und die Anwendung mit eingeschlossen. Es gibt die folgenden verschiedenen Kategorien von Kontexten: physischer, technischer, persönlicher, sozialer und anwendungsbezogener Kontext.

Kontextdaten können an den verschiedensten Stellen innerhalb einer verteilten Anwendung erhoben werden. Mögliche Kontextquellen sind Sensoren, Datenbanken, Anwendungen, Benutzereingaben und Interaktionen, die grundsätzlich verschieden, also heterogen, und verteilt sind. Heutige kontextabhängige Anwendungen und Szenarien sind nur denkbar, wenn die verschiedenen und verteilten Kontextdaten zusammengeführt werden können. Ein Kontextdienst hat die Funktion, Kontextdaten zusammenzufassen, zu interpretieren und der jeweiligen Anwendung zur Verfügung zu stellen. Für die Anwendung soll die Herkunft der Kontextdaten dabei weitestgehend transparent sein.

3.1.4 Gu's Ontology-based Context Model for Intelligent Environments

Gu et al. haben ein formales ontologiebasiertes Kontextmodell entwickelt und verwenden für die Darstellung die Web Ontology Language (OWL). Ziel ihrer Arbeit ist die semantische Kontextrepräsentation für die semantische Interoperabilität, das Reasoning über Kontext und den Austausch von Wissen (Knowledge-Sharing). Sie berücksichtigen die Klassi-

fizierung von Kontext, Abhängigkeiten zwischen Kontext und die Qualität von Kontext. Den Anwendungsbereich ihres Kontextmodells sehen sie in der Smarthome Environment, also in der Anwendungsumgebung intelligenter Häuser.

Die Ontologie soll dabei die große Vielzahl heterogener Kontextdaten erfassen können. So ist ihr Ansatz, eine allgemeine Anwendungsdomäne (upper ontology) zu definieren, die um spezifische Subdomänen (Domain specific ontologies) erweitert werden kann. Vorteil dieser Herangehensweise ist das dynamische Nachladen von zusätzlichen konkreten Domänen und eine damit einhergehende Entlastung der mobilen Endgeräte, die nicht zu jeder Zeit eine komplette Ontologie im Speicher halten müssen. Größere Ontologien entstehen so durch die Integration von Domänen-spezifischen Ontologien. Die Upper Ontology besteht aus den Basiskonzepten Person, Location, Activity und Computational Entity. Bei der Modellierung werden die Kontextdaten in direkten und indirekten Kontext klassifiziert. Direkter Kontext bezeichnet Kontextdaten, die entweder direkt gemessen (sensed) oder definiert werden. Indirekter Kontext entsteht durch Reasoning und Aggregation von direkten Kontextdaten.

Eine wichtiger Aspekt von Kontextdaten ist die Abhängigkeit zwischen einzelnen Kontextdaten. Diese muss modelliert werden, da dadurch Beziehungen zwischen Entitäten angegeben werden, und das ist ein Indiz für die Zuverlässigkeit und die Qualität von Kontextdaten. In dem Kontextmodell werden Kontextdaten durch Qualitätsattribute genauer spezifiziert. Dies ist wichtig, da z.B. gemessene Ortsinformationen innerhalb kürzester Zeit an Aktualität und somit an Wert verlieren können.

Es wird angenommen, dass definierte Kontextdaten eine höhere Zuverlässigkeit haben, als gemessene oder abgeleitete Daten. Anhand dieser Klassifizierung der Kontextdaten und in Abhängigkeit von der Zuverlässigkeit der unterschiedlichen Kontextarten wird über den Kontext geschlussfolgert. In Kombination mit den Qualitätsattributen der Kontextdaten können so Konflikte aufgespürt und beseitigt werden.

Dieser Ansatz basiert auf OWL und nutzt die spezifischen Vorteile von OWL hinsichtlich der Modellentwicklung und der Verarbeitung der Ontologien. Nachteilig an diesem Ansatz ist die Definition der Upper Ontology als Basis für die Ableitung der Lower Ontologies, da diese Herangehensweise die Allgemeingültigkeit und die Erweiterbarkeit einschränkt. Domänen-spezifische Konzepte der Subdomäne, die in der Upper Ontology nicht berücksichtigt wurden, können so nicht konsistent modelliert werden.

3.1.5 Chen's Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA)

Der Entwurf von Chen et al. (siehe [4]) wählt ebenfalls eine ontologiebasierte Herangehensweise. Es wird OWL für die Kontextmodellierung verwendet und es wird Reasoning auf Kontext unterstützt. SOUPA ist eine Weiterentwicklung von CoBrA-Ont, um Konzepte wie Document, Event und Policy, und somit Teil der Context Broker Architecture (CoBrA[1]). CoBrA ist eine Architektur für die Unterstützung von kontextabhängigen Systemen in der Intelligent Meeting Room Domain und bietet keine explizite Unterstützung für die Modellierung von allgemeinem Kontext in heterogenen Umgebungen. Die semantische

[1] `http://cobra.umbc.edu/index.html`

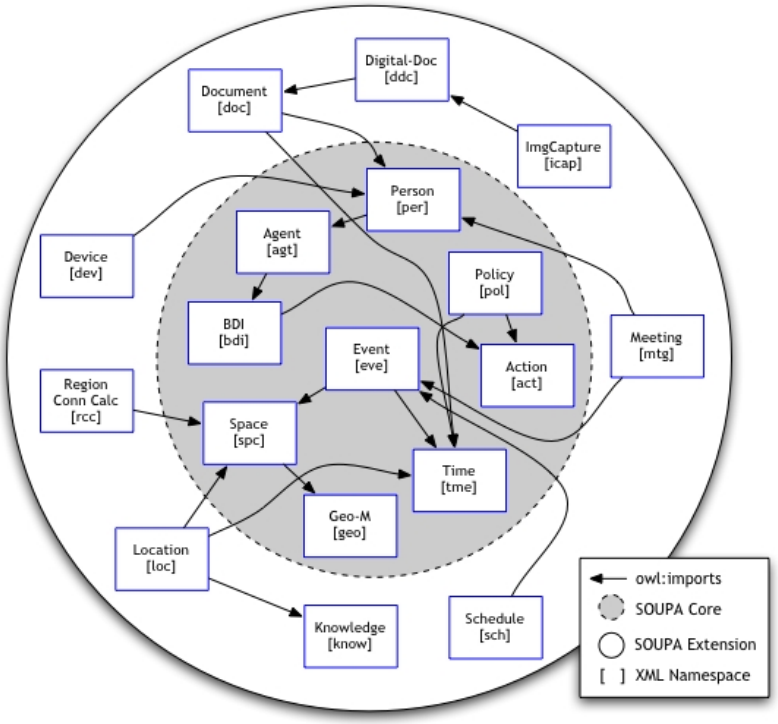

Abbildung 3.1: SOUPA Ontology 2004-06 [4]

Interoperabilität ist nur gewährleistet, wenn alle teilnehmenden Anwendungen die SOUPA Ontologie verwenden, beziehungsweise wenn die verwendeten Ontologien in SOUPA übersetzt werden.

In CoBrA wird OWL verwendet, um damit Ontologien zu definieren für die Modellierung von Kontext, die Erstellung gemeinsamer Vokabulare für den Wissensaustausch, für das Reasoning über die Beziehung zwischen Kontextdaten und das Domänenmodell und die Abbildung benutzerdefinierter Zugriffsregeln zur Sicherung der Privatsphäre des Nutzers. Der Context Broker sammelt Kontextdaten von verschiedenen heterogenen Quellen und integriert diese in ein einheitliches Modell, welches dann mit verbundenen Geräten geteilt werden kann. Der CoBrA-Ansatz unterstützt ressourcenschwache Endgeräte, die nur limitierte Fähigkeiten zum Sammeln und Reasoning von Kontextdaten besitzen und überträgt diese Aufgaben an einen zentralen ressourcenstarken Broker, der das Aufspüren von Kontext, Reasoning und das Auflösen von inkonsistentem Kontextwissen übernimmt. Durch die zentrale Datenverwaltung ist es einfach möglich, Schutz für die Privatsphäre und den Datenschutz zu implementieren.

SOUPA ist eine Menge von Ontologien, die den typischen Kontext in der Intelligent Meeting Room Domain modellieren soll. Es wurden dabei typische Konzepte und Beziehungen zur Beschreibung von physischen Ortsangaben, Zeit, Menschen, Software Agenten, Mobilen Geräten und Meetingevents definiert (siehe Abbildung 3.1). Auf diesen Konzepten kann Reasoning durchgeführt werden. Dabei kann unterschieden werden nach Reasoning über Ortsontologien, Geräteontologien und Zeitontologien. So lässt sich beispielsweise aus gemessenen Sensordaten, die ermittelt haben, dass sich eine Person in einem bestimmten Raum befindet und dem Wissen , dass dieser Raum Teil eines bestimmten Gebäude ist,

schlussfolgern, dass sich die Person in dem Gebäude befindet. Solche Aussagen können relevant sein, wenn Regeln für den Oberbegriff Gebäude definiert werden, und diese auch für alle Räume (Subkonzepte) in diesem Gebäude gelten sollen.

Um die Realisierbarkeit des CoBrA-Ansatzes im Easy Meeting System (Anwendung, die Intelligent Meeting Room Domain umsetzt) zu demonstrieren, wurde der zentrale Context broker als Agent implementiert, der auf der JADE Plattform (Java Bibliothek für FIPA Agenten) läuft und diesem Standard für Agenten entspricht. Für den Zugriff und die Bearbeitung der OWL Ontologien verwendet der Broker das Jena Semantic Web Toolkit [17]. Für das logische Schließen (Reasoning) wird Jess (Java-basierte Rule Engine)[9] verwendet. Der Vorteil von Jess gegenüber anderen verfügbaren Rule Engines ist die Interoperabilität mit existierenden Semantic Web Bibliotheken und Werkzeugen, die ebenfalls in Java geschrieben wurden.

SOUPA ist ein Ansatz mit dem Ziel, eine umfassende Ontologie für die Domäne des Ubiquitous Computing zu bieten. Damit haben Anwendungen in diesem Bereich Zugriff auf eine gemeinsame Ontologie und können kooperieren.

3.2 Verwandte Ansätze

Mit den folgenden Anforderungen werden die verwandten Arbeiten verglichen und analysiert. Dabei wird ermittelt, welche Anforderungen erfüllt beziehungsweise nicht erfüllt werden. Aus diesen Ergebnissen folgt die Motivation für diese Arbeit und es soll herausgestellt werden wo die Lücken sind und warum der hier zu entwickelnde Ansatz benötigt wird.

Anforderungen

- *Anwendungsdomäne*: Richtet sich der Ansatz wirklich an ältere Menschen? Werden die besonderen Anforderungen von dieser Zielgruppe unterstützt? Werden die hier beschriebenen Anwendungsfälle von den verwandten Arbeiten unterstützt? Zielstellung: älteren Menschen unabhängiges Leben zu ermöglichen!
- Einsatz von *Ontologien* zur Modellierung des Smarthomes (Smart environment), der Geräte und den Tasks
- Automatisch *Tasks* erkennen und ausführen, die der Absicht des Nutzers entsprechen.
- Wie erfolgt die *nahtlose Kooperation* von Geräten? Werden dazu Regeln eingesetzt? Wie?
- Können verschiedene heterogene Geräte dynamisch kooperieren um gemeinsamen einen Task auszuführen?
- *Nahtlose und dynamische Integration* heterogener Geräte: Anmelden, Abmelden, Bereitstellen von Funktionalität und Diensten (auch unbekannter Geräte)
- Welche Ressourcen können von dem dargestellten System integriert werden? Gibt es irgendwelche Einschränkungen?
- Hardwareanforderungen an die Geräte, auf denen System laufen soll und die Midd-

leware? Gibt es dabei Einschränkungen? Und welche?

- Context-Awareness Welcher Kontext muss vom System erkannt werden?
- Welche Schnittstellen müssen die Geräte implementieren?
- Technologien SOA, OSGi Wird eine SOA verwendet und warum?
- Welche Ereignisse werden erfasst?
- Einschränkungen

3.2.1 Interplay

Interplay ist eine Forschungsarbeit mit dem Titel „A Middleware for Seamless Device Integration and Task Orchestration in a Networked Home“ [18]. Motivation dieser Arbeit ist die für den Nutzer zunehmende Komplexität bei der Bedienung von vernetzten Unterhaltungsgeräten im Haushalt. Diese Geräte werden immer mächtiger und der Nutzer muss sich die komplexen Funktionalitäten merken, selbst um einfachste Tasks auszuführen. Für den Nutzer bedeutet dies zunehmend mehr Unbehagen und Frustration mit steigender Anzahl von Geräten, Inhalten und Daten.

Für die Lösung dieses Problems wurde eine Middleware entwickelt die heterogene Unterhaltungsgeräte integriert, Sitzungen verwaltet und dem Nutzer einfach erlaubt seine gewünschte Aktivität (Tasks) im Haus auszuwählen und auszuführen. Aus den Gerätefunktionalitäten, dem Ort, Inhalten und Nutzerpräferenzen im Haus kann Interplay dem Nutzer die möglichen Tasks zeigen, die er ausführen kann. Die Nutzerabsicht (user intention) wird aus einfachem Kontext geschlussfolgert und daraufhin können dem Nutzer Vorschläge unterbreitet werden.

Zielsetzung ist es die Absicht des Nutzers zu ermitteln und dann automatisch die geeigneten Tasks anzubieten beziehungsweise auszuführen. Dafür soll eine Middleware geschaffen werden, die es dem Nutzer ermöglicht seine Intention auszudrücken, die Gerätefunktionalitäten zu verstehen und aus den Gerätefunktionalitäten und Gerätefähigkeiten, Orten, Inhalten und Nutzerpräferenzen dem Nutzer mögliche Tasks anzubieten. Die Lösung soll auf existierender Unterhaltungsgeräten und der kommenden Jahre laufen.

Aus dem Anwendungsgebiet vernetzter Unterhaltungselektronik in einem Haus wurden folgende Aspekte extrahiert:

1. **Taskorchestrierung:** um einen Task auszuführen, müssen mehrere Geräte kontrolliert und konfiguriert werden.
2. **Task-Sitzung:** welche Geräte/Einstellungen sind oder waren zusammen an einem Task beteiligt. Dieser gemeinsame Zustand wird als *Session* bezeichnet.
3. **Gerätekollaboration:** Verwendung von verschiedenen möglicherweise inkompatiblen Geräten um einen Task auszuführen. Diese Geräte müssen nahtlos im Haus zusammenarbeiten.
4. **Auswahl von Inhalten:** Suchen und Finden der Inhalte, die über das gesamte Haus (Geräte und Verzeichnisse) verteilt sind.

Ermittlung der Nutzerabsicht

Beispielsweise schaltet der Nutzer den Fernseher ein und legt eine DVD ein, daraus wird dann geschlussfolgert, dass er sich die DVD anschauen möchte. Die Auswahl der möglichen Tasks wird von der Middleware Software auf einer einfachen pseudo-englischen Benutzerschnittstelle präsentiert und der Nutzer muss sich nicht mehr überlegen, wie ein Task ausgeführt wird *how to do a task* oder wo sich ein bestimmter Inhalt befindet, sondern nur *what to do* um einen Task auszuführen.

Beschreibung eines Tasks

Zwei technische Schlüsselkonzepte werden in Interplay entwickelt, die Pseudosätze und Tasksitzungen. In einem Pseudosatz besteht ein Task aus Verb, Subjekt und einem Target Device. Subjekt kann ein Inhaltstyp oder ein bestimmter Inhalt sein. Pseudosätze haben folgende Eigenschaften:

1. einfache und intuitive Methode um die Absicht des Nutzers zu beschreiben
2. können von Nutzer ohne Handbuch verstanden werden
3. genügend detailliert um in Gerätediensteaufrufe transformiert zu werden für die Taskausführung

Eine einfache Menüstruktur genügt, um dem Nutzer eine Auswahl an möglichen Tasks anzubieten, die seiner Intention am Besten entspricht. Ein solcher Task, den der Nutzer auswählen kann ist beispielsweise ['Play', 'Movie', 'LivingRoom DTV']. Die Tasks werden auf der Benutzerschnittstelle in einem leicht verständlichen Pseudo-English präsentiert (siehe Abbildung 3.2).

Abbildung 3.2: The Task Composition Screen

Das zweite Konzept Task-Sessions (siehe Abbildung 3.3) bezeichnet, dass ein Task in einer Session ausgeführt werden kann. Eine Session beinhaltet den Ausführungszustand verteilt über alle beteiligten Geräte (*spielt gerade*, *pausiert*). Aus den Pseudosätzen, die eine Aufgabe auf hohem Niveau beschreiben werden Geräteaufrufe erstellt. Somit ist dynamisches

Hinzufügen und Entfernen von Geräten in der Session möglich und keine komplizierte Logik in der Task Composition Procedure nötig.

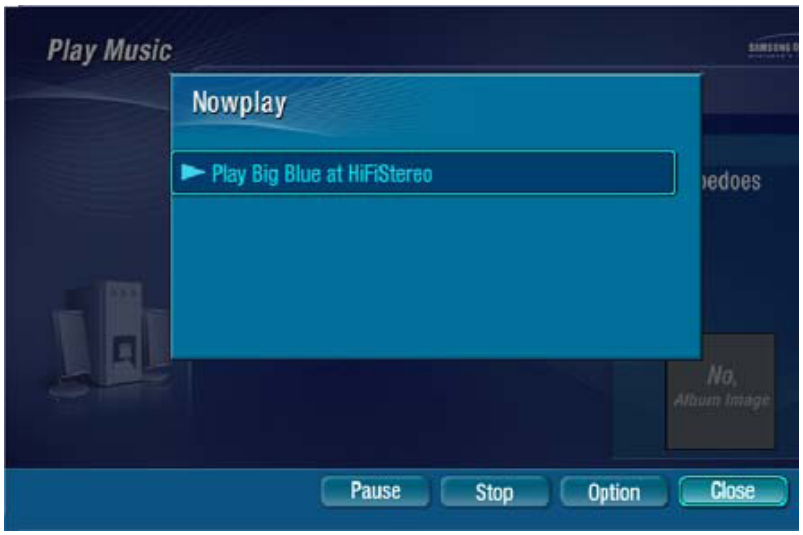

Abbildung 3.3: The Session Screen

Anwendungsfall

Die häufigsten Anwendungen aus dem Unterhaltungsbereich werden unterstützt. Dazu gehören Audio- und Video-Unterhaltung unter Nutzung von beliebigen Medienrenderern und Transcoding Geräten. Dann werden noch multimediale Anwendungen unterstützt wie das Drucken von Bildern, wobei es keine Rolle spielt, wo sich der Inhalt und das jeweilige Ausgabegerät befindet. Eine weitere Anwendung ist die Synchronisierung von Inhalten zwischen verschiedenen Speichergeräten.

Context-Awareness und Ontologien

Auf Nutzerkontext wird nicht eingegangen. Beachtet wird nur das Hinzukommen eines neuen Gerätes und dessen Funktionalität und Inhalte. Die Task- und die Gerätebeschreibung wird in RDF und OWL umgesetzt.

Task Beschreibung

Für die Task-Komposition werden funktionale Anforderungen (Requirements) mit verfügbaren Funktionalitäten der Geräte verglichen. Für die Beschreibung der Eingabe-und Ausgabetypen von Geräten werden Input/Output MIME type Spezifikationen (wie z.B. audio/mpeg content) verwendet. Geräte mit dem Eingabetypen *audio/mpeg* eignen sich so für den Task *spiele Audio ab*.

Von den Geräten werden folgende Parameter spezifiziert: der Typ, Name und die Beschreibung der jeweiligen Funktionalität. Als Kontextinformationen werden der Ort des Gerätes und die Gerätefähigkeiten genutzt. Für einen Fernseher sind das beispielsweise die Kontextinformationen: Fernseher -Bildschirmformat (Wide-Screen, Bildschirmgröße), stellt dar Audio/Video Video/Mpeg und Video/Dvd.

Darstellung eines Taskbeispiels, welches in der XML-Sprache RDF beschrieben ist.

```
<! -- Play Movie Task -->
<iplay:Task>
<taskVerb>PLAY</taskVerb>
<taskSubject>
   <Subject rdf:ID='Movie'>
    <requires rdf:resource='#AudioRenderer'/>
    <requires rdf:resource='#VideoRenderer'/>
   </Subject>
</taskSubject>
</iplay:Task>
<!-- Play Music Task -->
<iplay:Task>
<taskVerb>PLAY</taskVerb>
  <taskSubject>
    <Subject rdf:ID='Music'>
        <requires rdf:resource='#AudioRenderer'/>
    </Subject>
  </taskSubject>
</iplay:Task>
```

Implementierung

Der Prototyp wurde in Java implementiert auf einer prototypischen Fernseherplattform mit dem J2ME Foundation Profile. Für die Beschreibung der Geräte und Task Beschreibungsschemata wurde OWL und RDF verwendet. Die OWL/RDF Beschreibungen werden mit der Rule Engine JESS (Java Expert System Shell) [9] verarbeitet. Zur Unterstützung von UPnP-Geräten gibt es Plugins. Das Fernsehgerät ist die Kontrollstelle für das Auffinden und die Kontrolle der UPnP- Geräte. Jedes Gerät verwaltet seine Geräte und Task Beschreibungen, während der Auffindungsphase werden URLs dieser Beschreibungsdateien an das Fernsehgerät übermittelt.

3.2.2 AMIGO

Das AMIGO-Projekt [16] mit dem vollen Namen *Ambient Intelligence for the networked home environment* ist ein Projekt der Europäischen Union und fünfzehn der führenden Europäischen Industriepartner. Mehr Informationen dazu gibt es unter [2]. Start war im September 2004 und die Dauer des Projektes beträgt 42 Monate.

Amigo soll das volle Potenzial von Heimnetzwerken ausschöpfen, um so das Leben der Bewohner zu verbessern und soll dann in Richtung Ambient Intelligence gehen. In diesem Ansatz soll eine Middleware-Lösung entwickelt werden, die die bisher getrennten Domänen (Mobile Geräte, PCs, Heimautomatisierung, Unterhaltungselektronik) vereinigen. Bisher

[2] `http://www.hitech-projects.com/euprojects/amigo/project_information.htm`

wurde das Potenzial der Heimvernetzungstechnologie noch nicht entfaltet, da es kein konsistentes Framework für Heimsysteme gibt.

Hauptprobleme sind bisher:

- Die Usability des Systems (Geräte, Infrastruktur und Dienste).
- Der Mangel an genügend attraktiven Diensten. Dem Nutzer fehlt der Vorteil durch die Nutzung.
- Die Kosten sind hoch, da diese Technologien noch eine geringe Marktverbreitung haben.

Anforderungen an das System sind:

- Für die Usability muss die Benutzerschnittstelle einfach, vertraut, robust sein und den Datenschutz und Sicherheit einhalten.
- Interoperabilität, durch Middleware-Standards die funktionale Interoperabilität auf der Anwendungsebene liefern. Momentan gibt es noch keine Middleware für alle vier Domänen.
- Notwendig für eine leichte Installation: Automatisches Entdecken von Geräten und Diensten. Dienste müssen zusammengefügt werden können. Upgrade-Fähigkeit, Selbstadministrierung.
- Offene Middleware, die heterogene Dienste integriert und zusammenfügt. Dazu wird die Beschreibung der Dienste genutzt und die Erkennung und Komposition erfolgt bis auf ein Semantisches Level.
- AMIGO-Dienste müssen einen Vorteil über nicht vernetzte Dienste haben. Es wird Kontext über andere Geräte im System und den Nutzer gesammelt und daraus werden neue kontextabhängige Dienste entwickelt (Intelligent User Services).

Der Prototyp legt den Fokus auf die Anwendungsdomänen: Häusliche Pflege (Homecare), Sicherheit, Home information und Unterhaltung.

Kontextmodellierung und Reasoning

Zur Modellierung des Kontextes wird RDF verwendet. Nutzermodellierung und Profiling Kontextquellen sind Haushaltsgeräte, Komfortsensoren (Temperatur, Feuchtigkeit, CO2), Ortssensoren, Nutzeraktivitäteninformationsprovider. Der Informationsaustausch zwischen Kontextquellen und kontextabhängigen Diensten erfolgt in AMIGO über Webservice-Technologien.

Beispiel der Regel: *Maria möchte benachrichtigt werden, wenn ihre Kinder mit Freunden nach Hause kommen.*

Upon EnterTrue (Pablo.isAtHome) OR EnterTrue (Roberto.isAtHome) When (Pablo.isAtHome AND Pablo.withFriends) OR (Roberto.isAtHome AND Roberto.withFriends) Do Notify (Maria, „kids are home with friends“)

3.2.3 Nahtlose Integration von Ausgabegeräten

In dem Projekt [5] wird eine Infrastruktur vorgestellt, mit der multimediale Ausgabegeräte nahtlos in eine intelligente Umgebung integriert werden und die Ausgabe optimiert wird hinsichtlich der Gerätefähigkeiten und der momentanen Aufgabe des Nutzers. Die Koordination der Ausgabegeräte erfolgt basierend auf deren Selbstbeschreibung. Als Anwendungsbeispiel dient ein intelligentes Wohnzimmerszenario zur Validierung der Ergebnisse. Anforderungen an eine solche Infrastruktur sind die dynamische Verbindungsfähigkeit von Geräten, adaptive Benutzerschnittstellen und Darstellungen und die Darstellung von Ausgaben auf mehreren Geräten zur gleichen Zeit.

Die Infrastruktur basiert auf einer Schichtenarchitektur, mit der physischen Verbindungsschicht als Basis. Darauf setzt die abstrakte Verbindungsschicht auf, die sich um das dynamische Auffinden von Geräten kümmert, um die Organisation von Peers eines Channels in Gruppen und um die direkte Interaktion zwischen Peers. Die oberste Schicht ist die Koordinierungsschicht für die Selbstorganisation der Geräte. Es gibt keine zentrale Kontrolle und kein Wissen über andere Peers, so dass die Präsenz anderer Peers nicht geplant werden kann. Die Middleware ist physisch verteilt. Dadurch gibt es keinen Single-Point of failure. Jedes Gerät hat eine Instanz zur Middleware oder Verbindung zu einer entfernten Instanz via Proxy.

Es wird ein Modell benötigt für die Selbstbeschreibung der Geräte. Ein- und Ausgabegeräte gehören entweder die Kategorie Grafisch oder Akustik. Zur Identifikation der Geräte wird eine eindeutige ID verwendet. Für Geräte wie z.B. einen PC mit Eingabe- und Ausgabe-Komponenten werden beide Komponenten unter derselben ID registriert.

Für eine Eingabekomponente gibt es einen zuständigen Eingabedienst (siehe Bernsen's Taxonomy). Dieser liefert Informationen über den aktuellen Zustand des Eingabegeräts. Für eine Ausgabekomponente liefern Ausgabedienste Informationen darüber welche Inhalte unterstützt werden. Das sind zum Beispiel ASCII-Texte, die in Sprachausgabe ausgegeben werden können oder JPEG-Dateien, die grafisch ausgegeben werden. Bei der Grafikausgabe wird die aktuelle Auflösung berücksichtigt. Ein weiterer Komponententyp kann Inhalte nur generieren oder verarbeiten, aber nicht darstellen. Das ist zum Beispiel ein Landkarten-Renderer, der aus Koordinaten eine Landkarte generiert.

Der Zustand dieser Komponenten wird definiert durch Parameter wie die Sichtbarkeit, Bildschirmkoordinaten und die Fenstergröße. Bei dynamischen Ausgabekomponenten wird der aktuelle Zustand registriert wie: *spielt gerade*, *pausiert*, *gestoppt*. Nachteil dieses Ansatzes ist es, dass nicht ausreichend Informationen zur Unterscheidung von Diensten vorhanden sind, es fehlt die semantische Beschreibung der Dienste.

Aus diesem Modell wird abgeleitet, dass falls kein grafisches Eingabegerät vorhanden ist, dann kann der Nutzer keinen Button auf dem Display drücken.

Anforderungen an die Planung der Präsentation umfassen den aktuellen Zustand der Gerätekonfiguration, die geforderte Ausgabe und den aktuelle Stand der Ausgabe. Treten Änderungen in der Gerätekonfiguration auf dann müssen neue Ausgabetasks generiert werden. Der Kern des Präsentationsproblems sind kausale Abhängigkeiten, räumliche und zeitliche Beschränkungen. Es gilt einen Plan mit detaillierten Anweisungen zu erstellen für die finale Darstellung. Für die Lösung des Präsentationsproblems wird Künstliche Intelligenz

genutzt.

Dieser Ansatz unterstützt die spontane Verbindung zwischen heterogenen Geräten, die durch verschiedenste Netzwerke miteinander verbunden sein können. Durch die Selbstbeschreibung der Geräte, können diese dynamisch in die Gerätekonfiguration integriert und wieder ausgegliedert werden.

3.2.4 Dienstorientierte kontextbewusste Middleware

SOCAM (Service Oriented Context-Aware Middleware) (siehe [11]) wurde mit dem Ziel entwickelt Rapid Prototyping von kontextabhängigen Anwendungen zu ermöglichen. SOCAM konvertiert verschiedene physische Räume in denen Kontext erfasst werden kann in einen semantischen Raum, in dem kontextbewusste Anwendungen den Kontext teilen und darauf zugreifen können.

Architektur

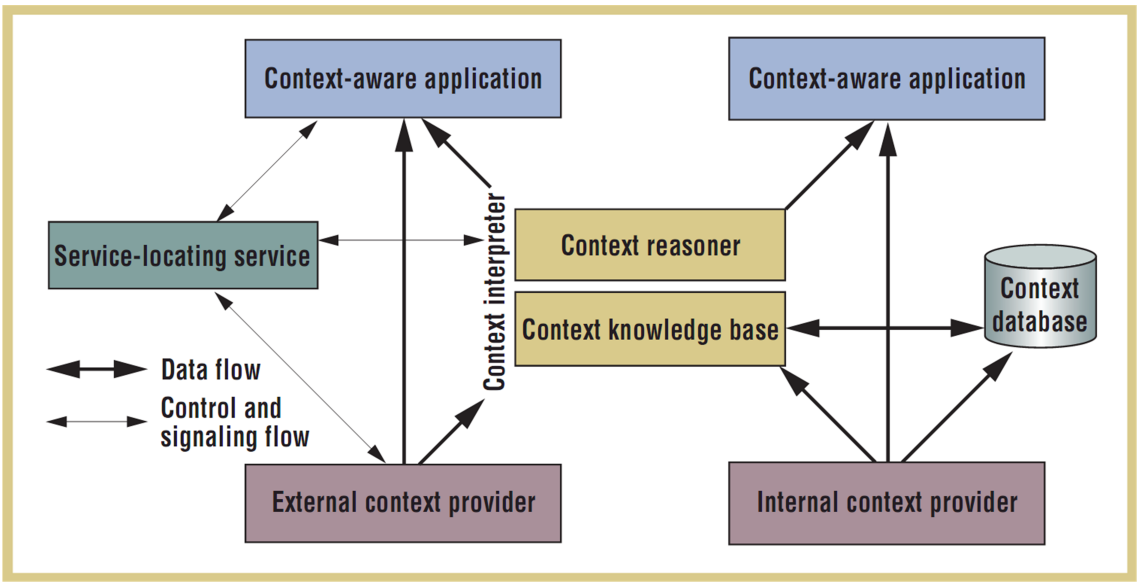

Abbildung 3.4: Überblick über die SOCAM Architektur [11]

Die Middleware besteht aus den folgenden Komponenten:

- Der Context provider abstrahiert nützlichen Kontext von heterogenen Kontextquellen (interne und externe) und wandelt diese in OWL-Darstellung um, so dass andere Dienstekomponenten diese teilen und wiederverwenden können. Eine externer Context Provider kann ein Wetterinformationsdienst sein, der Wetterdaten eines bestimmten Ortes liefert. Ein interner Context Provider sammelt Kontext von verteilten Sensoren, wie RFID Ortssensoren in einem Smarthome. Daraus können Ereignisse der Art *Person liegt auf dem Bett* oder *Person ist hingefallen* erzeugt werden. Nutzer und Dienste können entweder Kontext direkt abfragen oder Informationen über Kontextereignisse abonnieren. Der Context Provider nutzt ein Service Template oder einen OWL Ausdruck um zu spezifizieren welchen Kontext es liefert.

- Context interpreter bietet logische Schlussfolgerungsdienste für die Verarbeitung von Kontextinformationen an und ist somit ein Context Provider für High-level Kontext. Er besteht aus einem Context reasoner zum Ableiten der Kontextdaten und einer Kontextwissensbasis. Implementiert wird der Context Interpreter mit dem Jena2 Semantic Web Toolkit [3].

- Context database speichert Kontextontologien und vergangenen Kontext für eine spezifische Subdomäne. Jede Domäne hat eine logische Kontextdatenbank. Die Kontextdatenbank enthält T-Box Informationen über die Subdomäne und A-Box Informationen über die Instanzen der Kontextontologie.

- Kontextbewusste Anwendungen nutzen verschiedene Level des Kontextes und passen ihr Verhalten an den Kontext an.

- Service-Locating Service (SLS) bietet einen Mechanismus, der es dem Context interpreter und dem Context provider ermöglicht ihre Dienste anzubieten und Nutzern und Anwendungen diese Dienste zu finden. Eine Anwendung, die an einem Kontextwert interessiert ist schickt eine Anfrage (z.B. (John socam:befindetSichIn ?x)) an diesen Dienst und erhält als Antwort eine Referenz zu dem Context Provider, der diese Information enthält.

Abbildung 3.4 zeigt einen Überblick über die SOCAM Architektur. Alle SOCAM Komponenten sind als unabhängige Komponenten realisiert, die über heterogene Netzwerke verteilt werden können und miteinander interagieren können. Die Komponenten sind in Java implementiert und benötigen nur eine Java Virtual Machine in der sie laufen. Die Kommunikation zwischen den verteilten Komponenten erfolgt über Java RMI. SOCAM wird in die OSGi Diensteplattform integriert um damit ein zuverlässiges, sicheres System zu bauen, dass kontextabhängige Dienste in einem Smarthome anbietet. Jede Komponente wird als separates Bundle realisiert.

Kontextmodellierung

Da es schwierig ist Kontextwissen zentral zu verwalten und zu verarbeiten in einer Pervasive Computing Environment, wird hier ein hierarchischer Ansatz verwendet. Dabei werden die Kontextontologien in eine allgemeine High-level Ontology und domain-spezifische Ontologien unterteilt. Die High-Level Ontologie definiert die generellen Informationen über die physische Welt einer Pervasive Computing environment, dass sind Basiskonzepte von Person, Location, Computational Entity und Activity, wie auf Abbildung 3.5 zu sehen ist. Die Details dieser Basiskonzepte und deren Eigenschaften in jeder Subdomäne (Heim, Büro oder Fahrzeug) werden in domänenspezifischen Ontologien definiert. Dadurch wird die Menge an Kontextwissen erheblich reduziert und die Kontextverarbeitung vereinfacht.

Der größte Vorteil des hierarchischen Ansatzes ist, dass er Reasoning erlaubt. OWL wird verwendet, weil es ausdrucksfähiger ist als zum Beispiel RDF. Kontext wird hier in Prädikaten erster Ordnung dargestellt. Prädikat (Subjekt, Wert). Das Basismodell wird erweitert um boolsche Algebra (union, intersection, complement). So kann komplexes Wissen wie, die Essensvorlieben der Familienmitglieder beschrieben werden.

[3] `www.hpl.hp.com/semweb/jena2.htm)`

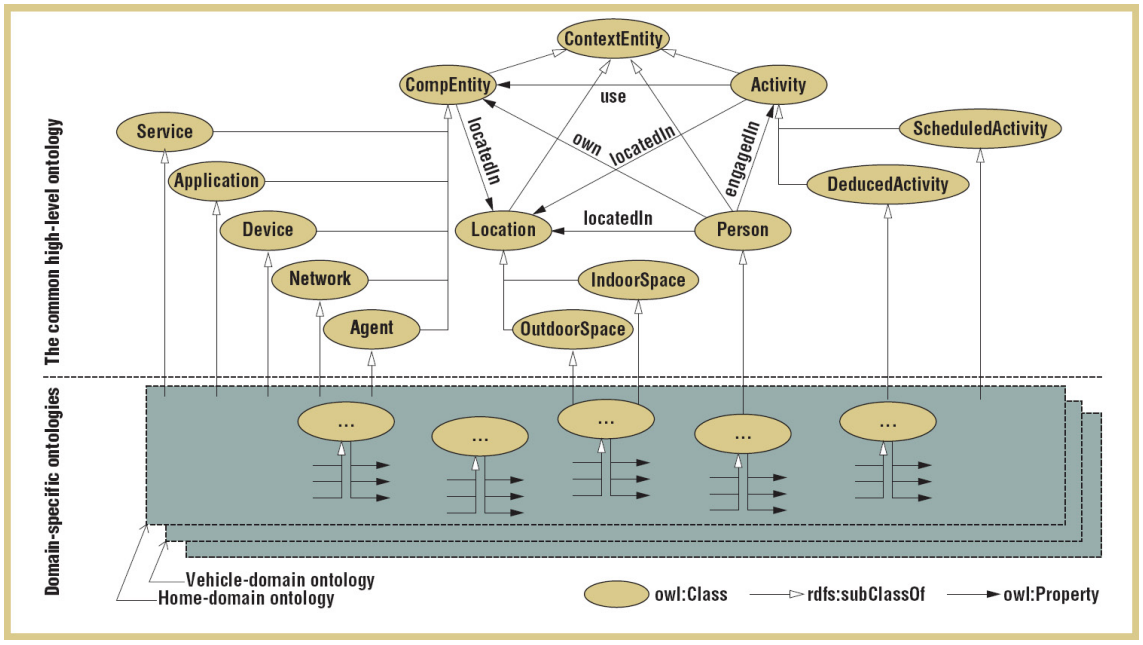

Abbildung 3.5: CONtext ONtology CONON [11]

Reasoning

SOCAM verwendet einen regelbasierten Ansatz basierend auf Logik erster Ordnung für das Reasoning auf Kontext. Unterstützt werden zwei Arten des Reasonings: Ontologiebasiertes Reasoning und vom Nutzer definiertes regelbasiertes Reasoning. Abbildung 3.6 illustriert Beispiele einer Teilregelmenge basierend auf einer Forward-Chaining Rule Engine.

(?user rdf:type Elderly) $\wedge$ (?user locatedIn Bedroom) $\wedge$ (?user hasPosture LieDown) $\wedge$ (Bedroom doorStatus Closed) $\wedge$ (Bedroom lightLevel Low) $\Rightarrow$ (?user status Sleeping)
(?user rdf:type Elderly) $\wedge$ (?user locatedIn ?room) $\wedge$ (TV locatedIn ?room) $\wedge$ (TV status On) $\Rightarrow$ (?user status WatchingTV)
(?user rdf:type Person) $\wedge$ (?user locatedIn Bathroom) $\wedge$ (WaterHeater status On) $\wedge$ (Bathroom doorStatus Closed) $\Rightarrow$ (?user status Showering)

Abbildung 3.6: Beispiele einer Teilregelmenge basierend auf einer Forward-Chaining Rule Engine

Kontextabhängige Dienste

Für jede Anwendung gibt es eine Kontextkonfigurationsdatei. Anwendungsentwickler definieren Regeln und bestimmen die Methoden, die aufgerufen werden, wenn die Bedingung wahr wird. Als Beispiele für kontextabhängiges Verhalten werden genannt: Gesundheitsfürsorge, Erinnerungshilfe und um Energie zu sparen. Die Regeln werden in einer Datei gespeichert und in den Kontextreasoner eingelesen. Während der Laufzeit können Regeln modifiziert und aktualisiert werden.

Anwendungsgebiet

Entwickeln einen Smarthome-Prototypen basierend auf der kontextabhängigen Infrastruktur. Zusätzlich wird noch eine Esszimmeranwendung entwickelt. Diese spielt Musik ab und passt das Licht an, wenn Familienmitglieder zusammen im Esszimmer essen. Die High-level Ontologie und Heim-Domänen Ontologie enthält auch Nutzerkontext wie den Geburtstag eines Familienmitglieds. Ortskontext wird durch Location Provider überwacht. Der Context Interpreter leitet die Raumaktivitäten ab (Abendbrot).

Zusammenfassung

Nutzen Ontologie-basiertes Kontextmodell in OWL für semantische Kontextdarstellung, Reasoning und Wissensaustausch. Basierend auf dem Kontextmodell wird eine Service-Orientierte Kontextabhängige Middleware (SOCAM) vorgeschlagen. Reasoning erlaubt es High-Level Kontext aus Low-Level Kontext, und impliziten Kontext von explizitem Kontext abzuleiten. Die Implementierung der Middleware erfolgt in der OSGi Implementierung von Prosyst. Damit werden die Komponenten-Bundles editiert, verwaltet, die Nutzerverwaltung durchgeführt und die Sicherheit kontrolliert.

3.2.5 DAFNE

DAFNE (Distributed Web-based Management Framework for Ambient Reconfigurable Services in the Intelligent Environment) [25] ist eine Framework-Lösung für intelligente Umgebungen und hat das Ziel heterogene Netzwerke und Geräte zu verwalten.

> Virtualisation of the ambient network infrastructure will enable the integration, interaction and interoperation of different devices from different vendors and diverse networking technologies.

Bisher fehlten Frameworks, welche die große Anzahl und Heterogenität von Geräten, Technologien und Ambient Intelligent Subsystemen vereinen um das Potenzial dieser Technologien auszuschöpfen und die Anforderungen einer intelligenten Umgebung zu befriedigen. Dabei soll solch ein System so unauffällig wie möglich die automatisierte Interaktion mit und zwischen Geräten ermöglichen. DAFNE ein wirklich integriertes intelligentes System, in der die Infrastruktur so unauffällig wie möglich ist und die Interaktion mit und zwischen Geräten autonom erfolgt. Dazu übernimmt DAFNE die Verwaltung von Anwendungen und Diensten und richtet sich dabei nicht nur an den Bereich des Smarthomes, sondern auch an den Firmen- und öffentlichen Sektor. Mit dieser dienste-orientieren Architektur lässt sich die Interoperabiliät, Verwaltung von Ressourcen und Rekonfiguration ermöglichen. Das System konzentriert sich hauptsächlich auf die Anwendungsfälle Gesundheitspflege älterer Menschen und Menschen mit Behinderungen und Sicherheit in Intelligenten Umgebungen.

Für die Umsetzung der virtuellen ambienten Netzwerk-Infrastruktur wird eine Kombination aus OSGi und UPnP verwendet. Dienste und Anwendungen werden auf diesem Framework als OSGi Bundles deployed. Das erlaubt die dynamische Registrierung, Aktivierung und Deaktivierung und das Auffinden von Diensten als auch die Interaktion zwischen Diensten von verschiedenen Anbietern. UPnP wird für die Kommunikation mit anderen Modulen der Intelligent Environment genutzt, dass heißt der Context-Awareness Komponente und der Ambient Network Infrastructure. Die einzelnen Anwendungen werden

als virtueller UPnP-Dienst deployed. Das Framework hat Mechanismen für die dynamische Integration von Komponenten, damit sich installierte Komponenten finden und zusammenarbeiten können um einen Task zu erledigen. Die Entwicklung und das Deployment von Diensten soll so dynamisch wie möglich sein, neue Dienste sollen wie bei Plug and Play hinzukommen und es soll eine dynamische Rekonfiguration des Systems erfolgen.

Anforderungen an intelligenten Umgebungen, die das DAFNE Framework erfüllt und die auch für die vorliegende Arbeit relevant sind.

1. Virtualisierung der ambienten Netzwerkinfrastruktur
2. Integration von innovativen Subsystem und Komponenten der Intelligenten Umgebung
3. Bereitstellung von fortgeschrittenen, personalisierten (angepasst an Nutzerpräferenzen), kontextabhängigen Diensten und Benutzerschnittstellen
4. Die einfache Entwicklung und Deployment, und koordinierter Ablauf der nutzerbezogenen Anwendungen
5. Orchestrierung und Koordination von Diensten und Anwendungen. Dadurch laufen diese koordiniert ab und es gibt keine Konflikte zwischen den Features.
6. Entwicklern werden High-Level Kommandos bereitgestellt.
7. Die Middleware ist kontextbewusst. Kontextereignisse der intelligenten Umgebung werden an die Anwendungen und Dienste weitergeleitet Anwendungen haben durch Virtualisierungschicht high-level Zugriff auf .
8. Schicht zur Erfassung von Kontext und der Situationsmodellierung durch Reasoning und ermöglichen von intelligenten Nutzerinteraktion.
9. Anwendungs und/oder Diensteanbieter sollen in der Lage sein existierende Dienste zu nutzen und zu kombinieren für komplexere Dienste. Dienste müssen sich selbst advertisen, beschreiben der Fähigkeiten, Attribute und Utilities.
10. Service Registry/Directory veröffentlichen und Auffinden von Diensten

3.2.6 Schlussfolgern auf Kontext im Smarthome: Ein SWRL Ansatz

Aktuelle Forschungsarbeit [20] aus dem Jahr 2007, dass ein kontextabhängiges System im Smart Home Bereich vorstellt. Der Ansatz basiert auf einer Service-orientierten Architektur (SOA). Es wird eine Intelligente Umgebung vorgeschlagen, die den Kontext und das Verhalten der Hausbewohner berücksichtigt um kontextabhängige Dienste anzubieten. Der Schwerpunkt dieser Arbeit wird auf auf das OWL-basierte Kontextmodell und auf die Schlussfolgerungschicht (Inference Layer) gelegt, die mit der SWRL Regelsprache umgesetzt wird.

Die Architektur zur Kontexterfassung besteht aus folgenden vier Schichten:

1. Wahrnehmungsschicht mit Sensoren zur Kontexterfassung

2. Kontextschicht stellt eine high-level semantische Sicht auf die ermittelten Daten bereit
3. Schlussfolgerungsschicht berechnet die Kontextdaten
4. Contextual Services Schicht bezieht sich auf Kontextdaten um Aktionen zu bestimmen.

Kontextmodell und logisches Schließen

Das Kontextmodell repräsentiert den Kontext zu einem bestimmten Zeitpunkt t und Ziel ist es den Kontext darzustellen, der auf Sensordaten basiert. Aus Gründen der Interoperabilität mit anderen Ansätzen wird eine OWL Ontologie zur Modellierung der Smart Home Domäne verwendet. Diese enthält die physische Repräsentation des Smart Homes (Türen, Wände, Fenster) und eine Repräsentation der Objekte, die in einem Haus vorkommen wie Möbel und elektrische Haushaltsgeräte. Ebenso wird der Bewohner im Kontextmodell repräsentiert, durch Modellierung seiner Charakteristiken wie Nutzerpräferenzen (Ort, Identifikation, etc.). Nach M. Kokar in [15] ist es unmöglich eine generische Ontologie für alle möglichen Anwendungsfälle zu modellieren. Es sollte eine eigene Ontologie mit anderen verfügbaren Ontologie erweitert und vereinigt werden (ontology alignment [7]).

Mit Ontologien erhält man eine formale Darstellung des Kontextes und somit können Methoden zum logischen Schließen auf den Kontextdaten angewendet werden [27].

Laut der Autoren ist das Neue an ihrem Ansatz, dass sie die Semantic Web Rule Language (SWRL) für das Schließen auf Kontextdaten verwenden. SWRL verfolgt als Ziel die Kombination von OWL und einer Schlussfolgerungssprache basierend auf RuleML. Mit SWRL kann eine verbreitete Sprache für das Kontextmodell (OWL) und das logische Schließen (SWRL) verwendet werden. Sie denken, dass logisches Schließen auf Kontextdaten noch nicht sehr entwickelt ist, aber sehr wichtig ist, um damit besser an den Nutzer adaptierte Dienste anzubieten.

Kontextabhängige Inferenzregeln

Mit der Möglichkeit des logischen Schließens kann eine Künstliche Intelligenz Schicht in die Systemarchitektur eingefügt werden und high-level Kontext aus low-level Kontext abgeleitet werden. Es gibt zwei Arten von Regeln:

- Backchain rules: Beschränkungen, direkt auf er OWL-Ontologie. Diese Regeln werden direkt in den OWL-Dateien beschrieben.
- Forwardchain rules: Mit Ausdrücken der Art (A→B) wird auf Axiomen geschlussfolgert.

Beispiele für Forwardchain-Regeln (Lichtsteuerung und Heizungssteuerung):

1. Wenn niemand im Raum ist, dann kann das Smarthome das Licht in diesem Raum ausschalten.
2. Wenn jemand den Raum betritt, dann kann das Smarthome das Licht in diesem Raum anschalten.

3. Wenn jemand zu Hause ist und die Temperatur eines Raumes kleiner ist als die Minimumtemperatur am Radiator in einem Raum, dann soll das Smarthome den Radiator einschalten.

Die Resultate der Inferenzschicht werden wie auch bei Sensoren an den ereignisgesteuerten Bus ausgegeben an den Akteure (z.B. gesteuerte Lampen, Heizungen) angeschlossen sind. Ein Akteur abonniert Ereignisse, auf die er dann entsprechend reagiert. Die generierten Ereignisse haben das gleiche Nachrichtenformat wie von Sensoren ausgelöste Ereignisse.

Implementierung

Zum Testen der Architektur wurde ein Testfall implementiert. Es wird die Robustheit des OSGi Ereignisgesteuerten Bus EventAdmin getestet, die teilweise Validität der Ontologie, den Schlußfolgerungmechanismus und die ausgelösten Aktionen. Der Prototyp basiert auf Knopflerfish, einer v4 OSGi Implementierung, welche EventAdmin integriert hat. Aus der Menge von möglichen regelbasierten OWL-Reasonern wurde der freie Reasoner KAON2 ausgewählt, da er javabasiert ist und SWRL unterstützt. Das Szenario orientiert sich am Follow-me Szenario [22] und ist wie folgt: Es bewegt sich jemand im Raum und das System schaltet automatisch das Licht an, welches dem Nutzer am nächsten ist.

In einem eingebetteten System werden, aufgrund der Datenmenge in der Ontologie, momentan noch fünf Sekunden Zeit für das Schlussfolgern benötigt.

Zusammenfassung

Dieser SOA-Ansatz hat seinen Schwerpunkt auf dem Kontextmodell und dem künstlichen Intelligenz Teil, der auf einer OWL-Ontologie und darauf schließenden SWRL Regeln basiert. Das Szenario konzentriert sich in den Beispielen auf Sensoren und Aktoren wie gesteuerte Lampen und Heizungen. Dabei stellt sich die Frage worin der Mehrwert einer solchen Lösung beseht und warum nicht einfach Bewegungsmelder und andere Sensoren genutzt werden?

Die SWRL-Regeln definieren Aktionen, die von Aktoren ausgeführt werden. Das System muss anhand von Daten in der Ontologie schlussfolgern welches die beste Aktion ist, die ausgeführt werden soll. Momentan ist die benötigte Zeit für das logische Schließen, aufgrund der Leistung von eingebetteten System, noch zu lang und dadurch könnte sich der weite Einsatz von Ontologien verzögern.

3.3 Heimvernetzungstechnologien

Es gibt verschiedene Technologien zur Heimvernetzung und diese arbeiten in ihren Bereichen gut. Sie haben sehr nützliche Funktionen zur Beschreibung von Geräten und ihren Diensten. Es mangelt allerdings bei allen an der Beschreibung wie Geräte kooperieren können. Von den gefundenen Lösungen setzt nur UPnP Regeln für die korrekte Kooperation zwischen Kontext und Objekten ein, aber diese Regeln sind zu statisch. So ist es schwierig umzusetzen, dass Geräte dynamisch kooperieren um neue Dienste zu schaffen, wenn diese nicht vorher programmiert wurden. Nachteilig ist auch, dass Geräte nut gefunden werden können, wenn es eine bekannte Beschreibung gibt.

> UPnP, Jini and HAVi have focused on connecting devices electronically in a variety of different forms that only provide simply device interactions and controls through the use of standardized APIs. [18]

HAVi eignet sich sehr gut für die Zusammenarbeit von Audio-Video Geräten, da es Videostreaming und die automatische Gerätesuche unterstützt. Es kann allerdings keine Befehle an einfache Geräte, wie Lampen oder Waschmaschinen schicken.

3.4 OSGi

OSGi steht für Open Service Gateway Initiative [4] und ist eine offene Architektur (offene Diensteplattform), die es ermöglicht eine große Anzahl an WAN Diensten in einem LAN (Smarthomes, Automobile) einzusetzen. Es ist ein leichtgewichtiges Framework zum Ausliefern und Ausführen von diensteorientierten Anwendungen und bietet Verwaltungsfunktionalitäten wie das Installation, Aktivierung, Aktualisierung und Entfernen von Diensten.

Die Vorteile von OSGi sind: Es ist eine Java-basierte Lösung, die nur eine Java Virtual Machine voraus setzt und ist somit plattformunabhängig (Hardware, Betriebssystem). OSGi bietet verschiedene Level an Systemsicherheit, von dem Signieren von heruntergeladenen Dienste bis Zugriff auf Objektkontrolle. Verschiedene Dienste von unterschiedlichen Anbietern können auf einer einzigen Gateway-Plattform gehostet werden. Desweiteren unterstützt OSGi eine Vielzahl von Heimvernetzungstechnologien. Die OSGi Spezifikation legt den Schwerpunkt auf die Bereitstellung von verschiedensten Diensten durch ein Gateway in Heim-, Büro- oder Fahrzeugumgebung. OSGi als herstellerunabhängige generische OSGi-Softwareplattform dient der Vernetzung einer Vielzahl von möglichen Geräten. Inzwischen wird OSGi in einigen Smarthome-Lösungen [5] und für Middleware-Lösungen verwendet. Das OSGi-Framework ist eine Service-Orientierte Architektur mit folgenden Eigenschaften

- geringe Kopplung,
- Komponenten finden sich dynamisch für Zusammenarbeit
- Komponenten können dynamisch an Geräte im Operationsfeld ausgeliefert werden
- Komponenten, die auf einer Vielzahl von Geräten, von sehr klein bis sehr groß laufen können

Das OSGi-Framework ist in folgende Schichten unterteilt.

- L0: Execution Environment
- L1: Modules
- L2: Life Cycle Management: fügt Bundles hinzu, die dynamisch installiert, gestartet, gestoppt, upgedatet und deinstalliert werden können.
- L3: Service Registry: bietet Kooperationsmodell für Bundles, welches Dynamik berücksichtigt. Es ist ein umfassendes Modell um Objekte zwischen Bundles zu teilen.

[4] `http://www.osgi.org/`
[5] Eine Auswahl ist zu finden unter `http://osgi.org/markets/smarthome.asp`.

Ereignisse werden beim Erscheinen und Verschwinden von Bundles ausgelöst.

Dienste werden durch Java-Interface spezifiert. Bundles implementieren dieses Interface und registrieren den Dienst bei der Service-Registry. Clients dieses Dienstes finden diesen in der Registry, oder reagieren darauf wenn er erscheint oder verschwindet. Das ist vergleichbar zur Service-Orientieren Architektur bei Web Services. Diese kommunizieren allerdings auf der Transportschicht und sind dadurch viel langsamer als OSGi Services, die direkte Methodenaufrufe verwenden.

Die Basis Software-Komponenten sind Service Bundles. Ein Bundle besteht aus dem Service Interface, einer Implementierungsklasse und einer Activator-Klasse. Das Service Interface legt fest, welchen Dienst diese Komponente bereitstellt und die Implementierungsklasse enthält die Implementierungsdetails. Die Activator-Klasse wird vom Framework genutzt zum Starten und Stoppen des Service-Bundles, und zum Registrieren und Deregistrieren von Services. Das OSGi-Framework ist die Laufzeitumgebung für die Komponenten. Es regelt den Lebenszyklus der Bundles, löst Abhängigkeiten zwischen Bundles auf, hat eine Registry zur Verwaltung der Dienste und bietet eine Kommunikationsplattform für die Kommunikation zwischen den Bundles.

Besonderes Merkmal ist die dynamische Umgebung, dass heißt OSGi Komponenten können zur Laufzeit heruntergeladen, installiert, aktualisiert, gestartet, gestoppt und entfernt werden, ohne dass das Gesamtsystem beendet werden muss. Die einzelnen Komponenten sind lose gekoppelt, dass heißt die Komponenten sind unabhängig voneinander und der Code eines Bundles ist vor Zugriffen eines anderen Bundles geschützt. Die Kooperation erfolgt durch das OSGi Connection Model.

Mit Package Connections werden statische Abhängigkeiten zwischen Komponenten bezeichnet. Service Connections sind dynamische Abhängigkeiten zwischen Komponenten. Probleme, die sich aus dem dynamischen Verhalten der Service Connections ergeben, werden durch das dynamische Verhalten von Bundles verursacht. Ein Bundle nutzt Services anderer Bundles und diese können zur Laufzeit verschwinden. Ein Bundle definiert mit Header in der Manifest-Datei, welche Pakete es import und welches es exportiert.

3.5 Dienste

Ein Dienst bietet einer Anwendung eine klar umrissene, in der Regel technische Funktionalität über eine Schnittstelle an [12].

Eine Dienste-orientierte Architektur zeichnet sich durch durch eine lose Kopplung zwischen den verschiedenen Diensten und einer hohen Kohäsion innerhalb eines Dienstes aus. Dienste arbeiten unabhängig voneinander. Die Funktionalität von Diensten ist gekapselt, man spricht auch von „Black-Box“. Die Kommunikation erfolgt über standardisierte Schnittstellen und Datenaustauschformate. Nach [3] ist *Kopplung* die Anzahl der Abhängigkeiten zwischen Subsystemen, und *Kohäsion* ist, ein Maß für die Anzahl der Abhängigkeiten innerhalb eines Subsystems. Ziel ist es die Abhängigkeiten zwischen verschiedenen Subsystemen zu minimieren. Die Kohäsion innerhalb eines Subsystems soll maximiert werden. Somit werden die verschiedenen Komponenten mit ihren Funktionalitäten weitestgehend voneinander entkoppelt und können unabhängig entwickelt und getestet werden.

Ein Service Provider bietet einen Dienst an und ein Service Consumer nutzt einen Dienst. Dazu stellt er eine Anfrage (Service Request) und erhält als Antwort eine Service Response. Hier eine Übersicht über die Vorgänge von Dienstanbieter und Dienstkonsument und deren Interaktion.

- Veröffentlichen (publish) eines Dienstes und Registierung (register).
- Finden eines Dienstes (find).
- Binden eines Dienstes (bind).
- Ausführen des Dienstes (execute).

Die hierarchische Zerlegung des Systems ergibt einen geordneten Satz von Schichten. Das System wird in Partitionen zerlegt und jede Partition ist für eine Klasse von Diensten zuständig. Beispielsweise wird das Softwaresystem in einem Auto in die folgenden Dienste zerlegt: Reisedienst, Präferenzdienst, Fahrzeugdienst [3]. Mit Orchestrierung wird die Anordnung von Diensten, in eine festgelegte Ausführungsabfolge, bezeichnet.

3.6 Zusammenfassung

Aus der Analyse der verwandten Arbeiten hat sich für mich ergeben, dass OSGi als Dienstepplattform sehr verbreitet ist und in bestehenden Smarthome-Lösungen verwendet wird. Ontologien eignen sich am Besten um die Konzepte einer Anwendungsdomäne zu beschreiben und soll in diesem Ansatz verwendet werden. Besonders das Forschungsprojekt AMIGO hat viele interessante Punkte, die mit meinem Ansatz übereinstimmen. Das Konzept welches ich aus den hier gewonnenen Kenntnissen entwickelt habe, wird im Kapitel 4 vorgestellt.

4 Konzeption

Für die Anforderungen, die während der Anforderungsanalyse ermittelt wurden (siehe Kapitel 2) soll eine Lösung entwickelt werden. In diesem Entwurf soll nicht auf Aspekte eingegangen werden, die von Middleware-Lösungen auf niedrigerem Level abgedeckt werden, wie Interoperabilität zwischen verschiedenen Geräten und Anwendungen, und Protokollen, Service Discovery, Sicherheit und Skalierbarkeit. Diese Arbeit legt den Schwerpunkt auf Softwaredienste in höheren Schichten, die auf basierenden Infrastrukturen aufbauen und den Nutzern Funktionalitäten zur Verfügung stellen [16].

Es wird die Vorgehensweise beschrieben, die zu dem hier dargestellten Konzept, geführt hat. Dabei werden verschiedene Lösungsansätze aufgeführt und begründet, wie sich die konkrete Entwurfsentscheidung durchgesetzt hat. Die Konzeption wird aus einer relativ abstrakten Sicht beschrieben, auf die Implementierungsdetails wird in Kapitel 5 eingegangen.

Dafür müssen die Funktionalitäten abstrahiert werden und eine Umwandlung in verschiedene Eingabe- und Ausgabemodalitäten möglich sein. Anhand der aktuellen Situation des Nutzers und seiner Umgebung soll ermittelt werden, welche Aktionen der Nutzer ausführen möchte.

Die Integration von Geräten vollzieht sich auf verschiedenen Schichten.

- In der Benutzerschnittstelle erfolgt eine Komposition der verfügbaren Funktionen, die von verschiedenen Geräten eingebunden werden.
- Die Funktionalitäten der einzelnen Geräte werden abstrahiert und für die Kooperation miteinander verknüpft.
- Die Situation des Nutzers wird erkannt und ihm werden mögliche ausführbare Aktionen angezeigt. Diese ergeben sich aus der Funktionalität der verfügbaren Geräte.

4.1 Regeln und Policen für die nahtlose Integration und Kooperation

Es wird eine dreischichtige Kooperationsarchitektur (siehe Abbildung 4.1) vorgeschlagen. Diese besteht aus den Policen (engl. Policies) auf der obersten Ebene, den Regeln (engl. Rules) auf der mittleren Ebene und Tasks auf der untersten Ebene. Der Nutzer sieht und konfiguriert nur die Policen auf der obersten Ebene, so dass sämtliche Einstellungen auf einer hohen Abstraktionsebene vorgenommen werden. Das System übernimmt die Übersetzung der Policen in Regeln und Tasks. Die Tasks definieren wie einzelne Dienste zusammengesetzt werden. Regeln definieren, wie Tasks ausgeführt werden. Die Policen, auf der höchsten Stufe, werden vom Nutzer parametrisiert und konfigurieren die Anwendung der Regeln. Je tiefer in der Hierarchie abgestiegen wird, desto größer ist die Flexibilität für das

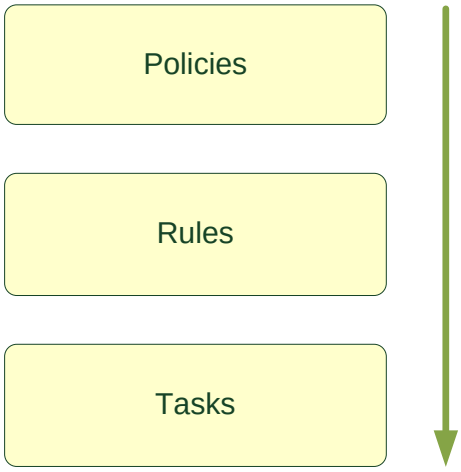

Abbildung 4.1: Dreischichtige Kooperationsarchitektur

Bilden von Regeln.

Der Nutzer befindet sich in einer Alltagssituation in seiner Anwendungsdomäne. Es tritt ein Ereignis auf, auf das reagiert werden soll, also eine Aktion ausgeführt werden muss. Das System erkennt die High-level Situation. Das System ermittelt dazu welche Geräte an dieser Situation beteiligt sind, und den dazugehörigen aktuellen Kontext. Der Nutzer möchte für diese Situation und dieses Ereignis eine Police festlegen. Dazu muss er wissen, welche Aktionen und mit welchen Geräten in dieser Situation möglich sind.

Damit der Nutzer Einstellungen vornehmen kann, muss ihm die Situation verdeutlicht werden. Das System zeigt dazu die beteiligte Geräte, den Kontext und auftretende Ereignisse an. Das System erzeugt automatisch mögliche Regeln und zeigt diese dem Nutzer in mehreren Detaillierungsgraden an.

Zum Beispiel: Es ist abends. Das Licht ist gedämpft und der Nutzer liegt auf dem Bett. Das System schlussfolgert aus diesem Kontext, dass sich der Nutzer entspannt und nicht gestört werden möchte. Es tritt ein Ereignis auf: Jemand klingelt an der Wohnungstür. Die Aktion ist: Zeige Videobild vom Eingang auf dem Fernseher an.

4.1.1 Policen

Policen sind höhere Regeln, die vom Nutzer auf hohem Level parametrisiert werden, um damit das Systemverhalten auf den tieferen Schichten zu definieren. Sie sind relativ starr und können nur das Systemverhalten der unteren Regelschichten anpassen und kontrollieren, wenn dieses von den Geräten angeboten wird. Der Nutzer definiert so was erlaubt ist und wie und was nicht.

> Policy is an emerging technique for controlling and adjusting the low-level system behaviors by specifying high-level rules [24].
>
> Police ist eine aufkommende Technik für die Kontrolle und Anpassung von low-level Systemverhalten durch das Spezifizieren von high-level Regeln. [24].

Es wird unterschieden zwischen gerätebezogenen Policen und nutzerbezogenen Policen. Gerätebezogene Policen haben, auf die Gerätefunktionalität bezogen, eine höhere Wertigkeit als die nutzerbezogenen Policen, dass heißt der Nutzer kann keine Werte einstellen, die vom System oder von einem Gerät nicht unterstützt werden. Anders ausgedrückt, der Nutzer legt das übergreifende Systemverhalten fest, dabei stützt sich das System aber auf die gerätebezogenen Policies. Nutzer parametrisieren somit die Bedienung, die zur Verfügung steht. Beispielsweise legt der Nutzer fest, bei bei welcher Temperatur sich ein Gerät einschaltet oder um welche Uhrzeit er geweckt werden möchte. Allerdings kann der Nutzer die Zentralheizung nicht auf 500 Grad Celcius schalten, sondern nur innerhalb der von der Zentralheizung vorgegebenen Grenzen. Der Gerätehersteller ist dafür verantwortlich die Grenzen für die einstellbaren Parameter des Gerätes zu definieren. Nutzerbezogene Policies werden über das Policy Interface in den HomeController eingegeben und in der Knowledge Base abgespeichert. Gerätebezogene Policen werden von den Geräteherstellern vordefiniert und bei der Anmeldung eines Gerätes, an den HomeController, in die Knowledge Base integriert.

Beispiele für gerätenahe Policen

- Jedes Gerät hat Public und Private Methoden (Funktionalität)
- Regelung des Systemverhaltens bei unvorhergesehene Systemzuständen. Es gibt einen minimalen Systemkern, der immer läuft.
- Verhalten bei einem Systemabsturz oder schwerwiegenden Fehler. Es gibt einen Neustart oder es gibt eine Startsequenz in der die Systeme hochgefahren fahren.
- Definierte Reaktion bei Abbruch
- Ordnung der Regeln. Systemkritische Regeln haben eine höhere Priorität als weniger kritische Regeln.
- Abarbeitung und Ablauf der Regeln wird durch deren Pritorät bestimmt.
- Nutzerereignis wie: Nutzer ist tot –> Wie erfolgt die Abarbeitung der offenen Aufgaben: FIFO, LIFO, Abbruch?
- Was tun bei Störfall? Ausnahme?
- Unerlaubter Zugriff
- Welche Geräte sind zulässig für Kooperation/Integration? Alle Geräte, die angemeldet wurden? Für welche Zeitdauer?
- Welche sind überhaupt verwaltbar? Bringt das Gerät seine Regeln mit, die von den Policen kontrolliert werden?
- Kontrolle der Daten, die zwischen den Geräten ausgetauscht werden.
- Kontrolle der Zugriffe auf Netzwerke.

Beispiele für nutzerbezogene Policen

- Während Kooperation Zugriffsbeschränkung auf relevante Funktionalität, es soll kein Auslesen von persönlichen Daten erfolgen.
- Ich will nicht, dass ich über den Fernseher telefoniere und dieser alle meine Telefonnummern kopiert oder meine SMS.
- Beispiel: Handy mit persönlichem Telefonbuch wird mit Fernseher und Telefonanlage oder Navigationsgerät integriert. Private Adressen sollen nicht sichtbar sein.
- Fremde Geräte, die nicht in den Haushalt gehören, sollen erst nach Nachfrage bei dem Nutzer und seiner Erlaubnis integriert werden.
- Fremde Personen, die das Haus betreten, sollen sich und ihre Geräte authentifizieren.
- Nutzer sagt *Ruhe* ich will nicht gestört werden! Wie können die Geräte das gemeinsam umsetzen, was muss jedes einzelne dafür tun? Der Nutzer kann das am HomeController einstellen, dass er jetzt nicht gestört werden möchte. Für nicht stören gibt es eine Police, die festlegt, dass Geräte keine akustischen Meldungen ausgeben. Ereignisse sollen dem Nutzer nur mitgeteilt werden, wenn sie wichtig sind. Der Anrufbeantworter soll Anrufe aufzeichnen und die Türklingel wird abgestellt. Für jede einzelne Geräteklasse ist festgelegt, wie sich ein Gerät dieser Klasse verhalten soll, um die Policy zu erfüllen. Kommt ein neues Gerät hinzu, wird die Policy gleich berücksichtigt.
- Fernsehen aufnehmen, wenn ich von Kontakten, die in meinem Adressbuch stehen, angerufen werde.
- Schützen der Privatsphäre.
- Vor um 8 Uhr morgens nehme alle Telefonate auf den AB auf.
- Wecke mich zu einer festgelegten Uhrzeit, zum Beispiel um 8 Uhr.
- Nutzer legt fest, dass er bei Statusmeldung nicht benachrichtigt werden möchte.
- Gerätesteuerung und Statusabfrage
- Statusmeldungen von Geräten, nur demjenigen zeigen, der Gerät bedient hat.
- Sobald jemand schauen will, ob zum Beispiel Waschmaschine frei ist, werden diese Statusmeldungen benötigt.
- Nur Eigentümer einer Todo-Liste darf diese modifizieren.
- Rechteverwaltung für Zugriff auf den HomeController, also die zentrale Steuerung.
- Rechteverwaltung für Zugriff auf Konfigurationen. Wer darf neue Regel anlegen: Eltern, Kind, Administrator. Ist von der Rolle abhängig. Altersbeschränkung der Inhalte.
- Wie lange sollen die Sitzungsdaten gespeichert werden; wenn überhaupt?
- Licht ausschalten, wenn der Nutzer schläft.
- Heizung herunterfahren, wenn der Nutzer in den Urlaub fährt.

- Policen, die Umweltbewusstsein umsetzen (Heizung und Licht nur dort, wo sich Nutzer aufhält)
- Schutz des Nutzers, durch Sicherung der Wohnumgebung
- Benachrichtigungsreihenfolge
- Nutzer definiert welche Ausgaben auf welchen Geräten angezeigt werden.

Definition von Policen durch den Nutzer

Dem Nutzer werden Policen auf der Benutzerschnittstelle des Home Controllers angezeigt. Dazu werden zu jeder Police die verfügbaren Parameter und die dazugehörigen Wertebereiche dargestellt. Der Nutzer wählt aus diesen die gewünschten Parameter und Werte aus. Die Policen werden in einem einheitlichen Format dargestellt, sind also unabhängig von einer Herstellerdefinition.

Eine andere Variante ist es, dem Nutzer nach einem Ereignis, welches eine Bearbeitung erfordert mögliche Reaktionen anzubieten und der Nutzer wählt eine Variante aus. Darüber hinaus kann der Nutzer festlegen, wie sich das System in Zukunft in dieser Situation verhalten soll.

Der Nutzer legt mit einer Police das Systemverhalten als Reaktion auf ein Ereignis fest. Die Police wird mit der Komponente Police-zu-Regel-Übersetzer auf die darunter liegenden Regeln abgebildet.

Wenn ein neues Gerät in die Anwendungsdomäne einfügt wird, dann muss das Verhalten des Systems für diesen Fall geklärt sein. Der Nutzer muss sagen, wie das Gerät eingeordnet wird. Der Nutzer kann auch festlegen, wie sich das System beim nächsten Mal verhalten soll, wenn ein neues Gerät hinzugefügt wird. Damit kann das Standardverhalten für diese Situation definiert werden.

Policen werden zerlegt in Regelbestandteile, die an eine Syntax gebunden sind. So erfolgt die Abbildung von Policies auf Regeln.

4.1.2 Regeln

Regeln befinden sich in der Kooperationsarchitektur auf der mittleren Ebene. Sie sind detaillierter als Policen und können in verschiedenen Regelsprachen ausgedrückt werden. Für die Middleware ist es nicht entscheidend in welcher Regelsprache die Regeln umgesetzt werden, da es die übergeordneten Policen gibt. Regeln bestimmen wie etwas gemacht werden soll. Tasks bestimmen was exakt gemacht werden soll.

Regeln definieren die Ausführung von Tasks. Regeln sind abhängig vom Kontext des Nutzers. Für jede Domäne gibt es eine minimale Regelmenge, die vorhanden sein muss, um das Systemverhalten zu definieren. Die Grundregeln für Standardtasks sind vordefiniert. Der Administrator kann zusätzlich Regeln definieren, wenn es neue Geräte mit neuen Tasks gibt und Abhängigkeiten zwischen verschiedenen Tasks geregelt werden sollen.

Das Systemverhalten wird definiert durch die Beschreibung der Regeln mit einer Regel-

sprache, die dem Event-Condition-Action (ECA) Pattern folgt (z.B. RuleML). Jede Regel besteht hierbei aus einem Ereignis-Teil, einem Bedingungs-Teil und einem Aktions-Teil. Zwischen Ereignissen und Kontextdaten wird hier nicht unterschieden. Wenn von einem Gerät ein Ereignis ausgelöst wird, dann wird es als eine Kontextänderung betrachtet. Das Ereignis wird von einem Gerät ausgelöst.

In diesem Pattern wird durch das Ereignis ein Interesse modelliert (z.B. an einem Wechsel in den Kontextdaten). Condition steht für eine Bedingung, die vor Ausführen der Aktion erfüllt sein muss. Die Aktion wird ausgeführt, wenn der Bedingungsteil der Regel erfüllt ist.

4.1.3 Tasks

Die Tasks befinden sich in dieser Hierarchie auf unterster Ebene und ein Task definiert eine atomare Aufgabe, die ausgeführt werden kann. Atomare Tasks, sind ganz einfache Tasks, die die Funktionalität von genau einem Dienst repräsentieren. Komplexe Tasks sind aus mehreren atomaren Tasks zusammengesetzt. Ein Beispiel für einen atomaren Task ist der Task *Schalte Licht An*. Ein komplexer Task ist der Task *Nehme Video Auf*, der die Dienste des ContentProviders und den Dienst CaptureDevice beinhaltet. In der Ontologie beschreibt das Konzept eines Tasks eine Kooperationsaufgabe zwischen Geräten.

Tasks ergeben sich aus den verfügbaren Diensten, die zusammengesetzt werden, um die Kooperation zwischen den Geräten zu realisieren. Ein Task ist eine Zusammensetzung von mehreren Diensten und definiert die Funktionalitäten, die sich aus der Gerätekonfiguration und der aktuellen Situation des Nutzers ergibt.

Die Darstellung der Tasks erfolgt durch Beschreibung dieser Konzepte in der Task-Ontologie. Durch Klassifizieren auf den Konzepten der Ontologie mit einem Reasoner werden die möglichen Tasks, die sich aus den Gerätefunktionalitäten ergeben, automatisch gefunden. Die Erzeugung eines Tasks, wie ein Task mit Parametern initialisiert wird und die Ausführung eines Tasks wird in Unterabschnitt 4.2.4 beschrieben.

4.2 Architektur

An dieser Stelle wird der Entwurf für die Architektur vorgestellt. Zuerst wird die Architektur im Überblick vorgestellt, dann werden die einzelnen Komponenten mit ihren Aufgaben vorgestellt und zum Schluss wird das Zusammenspiel und der Signalfluss zwischen den Komponenten beschrieben. Das System ist eine Anwendungsorientierte Middleware, und befindet sich in der Anwendungsebene des OSI-Referenzmodelles.

Entsprechend der Anforderung *Nutzen von existierenden und verfügbaren heterogenen Geräten (PCs, Unterhaltungselektronik, Haushaltsgeräte, Mobile Geräte)* muss die Architektur für ein Verteiltes System entwickelt werden, welche verschiedene heterogene Geräte integriert.

Es wird eine Dienste-orientierte Architektur gewählt (siehe Abschnitt 3.5). Die einzelnen Geräte werden als Dienste realisiert und können so lose gekoppelt werden. Die Kommunika-

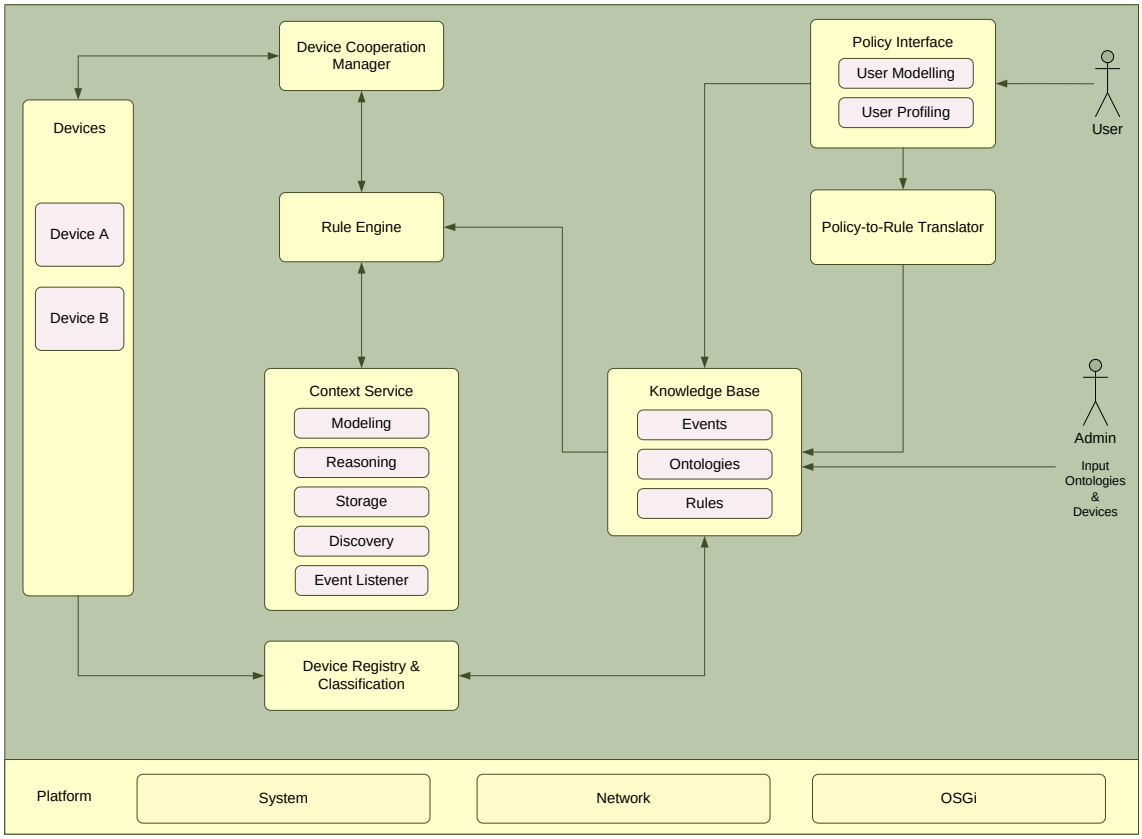

Abbildung 4.2: Architektur

tion zwischen den Diensten erfolgt über Web-Service-Technologien. Ein Gerät, beziehungsweise eine Komponente, unterscheidet sich von anderen Geräten, durch die Dienste die es anderen Subsystemen bereitstellt. Dienste sind Software-Komponenten, welche eigentlichen physischen Geräte vor dem Nutzer verbergen. Dem Nutzer wird die Gerätefunktionalität über benutzerfreundliche Schnittstellen angeboten, über die er die Geräte bedient.

Die Dienste laufen auf einer Diensteplattform (siehe Abschnitt 3.3). Für die Umsetzung wird eine OSGi-Implementierung verwendet (siehe Kapitel 5). Dazu wird jedes Gerät als Bundle realisiert, wodurch eine einfache Verteilung realisiert wird. Es gibt einen zentralen HomeController, auf dem die Kernfunktionalität der Komponenten installiert wird. Auf alle Geräte, die in der Anwendungsdomäne vorkommen können und in das System integriert werden sollen, wird der Teil der Middleware installiert, der für die Kommunikation mit dem HomeController notwendig ist. Geräte, auf denen die Middleware nicht direkt installiert werden kann, da sie nicht die Hardware-Voraussetzungen erfüllen, werden durch Technologien wie Jini Surrogate[1] gekapselt. In Abbildung 4.2 ist der Entwurf der Systemarchitektur mit den konzipierten Komponenten dargestellt.

Es gibt eine Unterscheidung zwischen physischen Geräten und logischen Komponenten. Auf einem Gerät können mehrere logische Komponenten installiert werden. Es ist ein verteiltes System und die einzelnen Komponenten können aus Performanz, Sicherheitsgründen auf verschiedene Hardwareknoten verteilt werden.

[1] `https://surrogate.dev.java.net/`

Auf den Geräten in der Smarthome-Umgebung läuft dieses System. Die Geräte stellen dem System ihre Funktionalität als Dienste zur Verfügung. Ein Gerät meldet für jede abgrenzbare Funktionalität einen Dienst an. Das bedeutet, dass zum Beispiel ein Videorekorder einen Dienst *Video abspielen* und einen weiteren Dienst *Video aufnehmen* anmeldet. Die Middleware läuft auf einer Virtuellen Maschine und kann somit auf allen Geräten installiert werden, die diese Virtuelle Maschine unterstützen.

4.2.1 Home Controller

Der Home Controller ist das zentrale Gerät mit dem der Nutzer interagiert. Auf ihm werden die Komponenten installiert, die für die zentrale Verwaltung und Kontrolle der beteiligten Geräte und deren Funktionalitäten, in der Anwendungsdomäne, zuständig sind. Er wird nur einmal auf einem zentralen Gerät in der Anwendungsdomäne installiert und ausgeführt, da so die Verwaltung der verteilten Geräte und Funktionalitäten am Einfachsten erscheint. Mit dem Home Controller erhält der Nutzer Zugriff und die Kontrolle auf alle angeschlossenen Heimautomatisierungsgeräte.

Der Zugriff auf diese zentralen Funktionen ist von jedem Standort, Computer oder anderen geeigneten Gerät dieser Anwendungsdomäne möglich, also auch per Webschnittstelle durch ein beliebiges an das Internet angeschlossene Endgerät wie Mobiltelefone, PDA, Webconsolen.

Der Home Controller übernimmt die Aufgaben eines Routers und bietet diverse Kommunikationsschnittstellen zu Heimvernetzungstechnologien (siehe Abschnitt 3.3) und zu Web-Services.

Als zentrale Instanz ist der Home Controller dafür verantwortlich, Geräte zu finden und zu registrieren. Eine Netzwerkverbindung ist für die Vernetzung der Geräte zwingend erforderlich. Geräte, welche in die Umgebung hinzukommen, auf denen die Middleware läuft und eine Netzwerkverbindung besteht, suchen bei der Initialisierung nach dem Home Controller. Das erfolgt wie bei Peer-to-Peer-Netzwerken über Discovery-Protokolle. Eine Suchanfrage wird per Broadcast in das Netzwerk gesendet. Entweder wird dann der Home Controller auf direktem Weg erreicht und dieser schickt eine Antwortnachricht zurück an den Sender, oder ein anderes Gerät in Reichweite, welche bereits beim Home Controller registriert ist, antwortet auf die Suchanfrage. Dieses Gerät arbeitet als Proxy und schickt die Adresse des Home Controllers als Antwort zurück. Wenn das suchende Gerät, die Home Controller-Adresse erhalten hat, stellt es eine direkte Verbindung zu diesem her. Wenn die Kommunikationsverbindung aufgebaut ist, übermittelt das Gerät seine Beschreibungsdaten zu Gerätedaten, zur Funktionalität und den Geräte-internen Policy an den Home Controller. Der Home Controller registriert das Gerät in der Device Registry (siehe Unterunterabschnitt 4.2.14.1) und trägt die Policies in der Knowlegebase ein.

Benutzerschnittstelle

Der Home Controller hat eine übersichtliche Benutzerschnittstelle, die den zentralen Zugriff auf die Konfiguration von Geräten, Policen, Regeln ermöglicht. Die Benutzerschnittstelle ist in drei verschiedene Sichten unterteilt, die die den Zugriff auf Einstellungen und Funktionen regulieren. Es gibt eine Sicht für den Nutzer, eine für den Administrator und eine für

den Anwendungsentwickler. Je nach Wünschen des Anwenders können sich die Sichten überschneiden. In den folgenden Paragraphen werden die Sichten beschrieben und inwieweit sie sich voneinander unterscheiden.

Nutzersicht

Die Nutzersicht ist die Sicht, über die der Nutzer mit dem Home Controller interagiert. Darüber erfolgt die Kontrolle und Konfiguration von Geräten. Der Nutzer definiert damit Policies, welche die Kooperation und Integration von Geräten festlegen. Dafür wird auf die Komponente Policy Interface zugegriffen, welche die Eingabe des Nutzers verarbeitet und mit Hilfe des Policy-To-Rule-Transformers in Regeln transformiert beziehungsweise direkt in der Knowledgebase abspeichert.

Der Nutzer meldet sich beim Home Controller an und kann sich den Status der vorhandenen Geräte in seiner Wohnumgebung anschauen. Dazu gibt es eine Statusampel, die mit den Farben grün, gelb und rot den Gesamtzustand des Systems visualiert. Wenn die Statussampel grün anzeigt, dann ist der Status aller Geräte in der Wohnumgebung in Ordnung. Bei gelb gibt es Warnungen, die beachtet werden sollten. Bei rot ist ein Fehler aufgetreten, der behandelt werden muss. Der Nutzer kann sich den Fehler anzeigen lassen und erhält Lösungsvorschläge, wie der Fehler behoben werden kann. Für jedes Gerät hat der Gerätehersteller definiert, was der Normalzustand ist, ein Zustand bei dem eine Warnung ausgelöst werden soll und was ein Fehlerzustand ist. Der Nutzer kann sich zu jedem Gerät anzeigen lassen, wie der Zustand definiert ist. Desweiteren hat er hat die Möglichkeit, innerhalb von herstellerseitig definierten Grenzen, für ein Gerät den Normalzustand, Warnungszustand und Fehlerzustand zu definieren.

Der Nutzer hat unterschiedliche Möglichkeiten, um die Kooperation zwischen Geräten zu regeln. Mit Policies definiert der Nutzer das Verhalten des Systems in Bezug auf das Kooperationsverhalten und die Integration von Geräten auf hohem Niveau. Die Benutzerschnittstelle bietet die Funktionalität zum Hinzufügen eines neuen Gerätes. Beim Hinzufügen werden die Konfigurationsmöglichkeiten eines Gerätes, mit den Anderen in dieser Umgebung, angezeigt. Der Nutzer wählt aus der angezeigten Liste mit potentiellen Geräten für die Kooperation die gewünschten Geräte aus. Damit wird die Kooperation der Geräte auf der höchsten Ebene definiert. Eine andere Möglichkeit ist es, dem dem Nutzer in einer gegebenen Situation, alternative Handlungsvorschläge zur Erreichung eines bestimten Ziels anzubieten, welches die Kooperation von Geräten einbezieht. Der Nutzer entscheidet sich für einen Handlungsvorschlag und aus dieser Entscheidung werden Policen für die Kooperation von Geräten generiert.

Für die spezifischeres Anpassung der Policies an abweichendes Verhalten werden diese in separate Profilen unterteilt. Diese Profile sind kontexabhängig und stehen in Zusammenhang mit der Situation des Nutzers, anwesende Personen, Gesundheitszustand des Nutzers wie Krankheit, geplante längere Abwesenheit (Urlaub), der Tageszeit.

Angenommen, der Nutzer erhält spontanen Besuch von seiner Familie, dann will er die normale Ausführung der Regeln unterbrechen. Der Nutzer ruft dazu den Home Controller auf und wechselt von der Standardausführung der Regeln auf das Profil *jemand ist zu Besuch*. Wenn das installierte System in der Wohnumgebung Kontext ermitteln kann, um solche Situation selbständig festzustellen, dann kann der Wechsel zwischen den Profilen automatisiert erfolgen. Der Nutzer hat aber dabei immer die Möglichkeit, das aktuelle

Profil schnell zu wechseln und die reguläre Ausführung zu unterbrechen.

Der Nutzer kann sich alle existierenden Profile und Policen im Home Controller anzeigen lassen. Er hat die Möglichkeit folgende Aktivitäten für Tasks und Policen durchzuführen.

- Neue Profile und Policen definieren
- Bestehende Profile modifizieren
- Löschen eines Profils
- Unterbrechen der Ausführung
- Anhalten und Fortsetzen der Ausführung

Wie Geräte miteinander kooperieren dürfen, hängt von folgenden Parametern ab.

- Wer ist der Eigentümer und wer ist der Nutzer des Gerätes?
- Wer darf das Gerät benutzen und wann, in welchem Kontext?
- Wo befindet sich das Gerät?
- Was ist die Funktion des Gerätes?
- Welche Funktionen dürfen verwendet werden?
- Welche Inhalte stellt es dar?
- Wird ein Gerät gerade verwendet? Kann es dann trotzdem für eine andere Aufgabe genutzt werden?
- ...

Der Nutzer hat in dieser Sicht Bausteine, aus denen er die Policen zusammensetzen kann. Es werden ihm die einzelnen Geräte visualisiert, deren Standort angezeigt, mit welchen anderen Geräten diese kooperieren können.

In dem System sollen die Geräte so intelligent sein, dass sie sich weitestgehend dem Bewusstsein des Nutzers entziehen. Dazu muss das System ermitteln können, welche Entscheidungen es dem Nutzer abnehmen kann und welche Entscheidungen der Nutzer ausschließlich alleine treffen muss. Ziel ist es die Balance, zwischen Abstraktion der Geräte und ihren Funktionalitäten und der Kontrolle durch den Nutzer, zu finden.

Die Integration eines neuen Gerätes vollzieht sich folgendermaßen. Der Nutzer möchte ein neues Gerät integrieren. Dem Nutzer wird auf dem Home Controller angezeigt, dass ein neues Gerät erkannt wurde. Es werden die Funktionalitäten des Gerätes angezeigt und wie dieses mit dem bestehenden Geräten kooperieren wird. Nachdem ein Gerät zur Domäne hinzugefügt wird, kann es ab diesem Zeitpunkt vom System verwendet werden.

Administratorsicht

In dieser Sicht nimmt der Administrator die Einstellung am System vor. Der Administrator definiert feingranulare Regeln vor, die von den Policen in der Benutzersicht verwendet werden. Der Administrator legt fest, wie neue (Input-/Output-/Verarbeitungs-) Geräte

dynamisch gefunden werden. Dazu wird die existierende Ontologie, für jedes neue Gerät, um ein Konzept erweitert. Der Administrator hat als einzige Person direkten Zugriff auf die Ontologien. Er erweitert die Regeln für die Kooperation zwischen den Geräten.

Wenn ein Gerät nicht automatisch in die Ontologie eingeordnet werden kann, dann schaut sich der Administrator die Funktionalitäten des Gerätes an und fügt dieses manuell ein. Der Standardfall sollte aber sein, dass jedes auf dem Markt erhältliche Gerät nahtlos in das System integriert werden kann, ohne dass eine weitere Konfiguration erforderlich ist. Wenn es bisher in dieser Umgebung keinen VideoRenderer gegeben hat, das Konzept unbekannt war, dann wird definiert, dass ein VideoRenderer ein Gerät ist, welches Inhalte vom Typ Video darstellt. Falls es das Konzept für den Inhaltstyp Video bisher noch nicht gab, dann wird das Konzept Video unter darstellbaren Inhalten eingeordnet.

Das System ermittelt in diesem Augenblick wie das neue Gerät mit den existierenden Geräten kooperieren kann. Wird ein Videorekorder eingefügt, dann ermittelt das System, dass der Videorekorder Inhalte des Typs Video von VideoProvidern aufnehmen kann. Neue Konzepte für den Videorekorder sind Gerät VideoCapturer/VideoProvider, Inhalt Video, Task aufnehmen.

Der Administrator definiert Sicherheitsregeln für den Einsatz von Geräten. Auf Sicherheitsaspekte soll in dieser Diplomarbeit nicht näher eingegangen werden, aber es soll dennoch erwähnt werden, dass fremde Geräte ein potentielles Risiko darstellen, wenn nicht genau definiert ist, welchen Code sie ausführen dürfen und von welchem Besitzer sie stammen. Besonders in Anbetracht der aktuellen Meldungen, die von immer mehr Viren uns Sicherheitsschwachstellen in Betriebssystem berichten, muss in diesem sensiblen Bereich alles dafür getan werden, dass Angriffe absolut unmöglich sind.

Der Administrator führt Software-Updates durch, wenn diese nicht automatisch durchgeführt werden können. Dazu gehört auch die Aktualisierung von Ontologien, um neue Konzepte und Regeln, die von anderen Anbietern stammen. Der Administrator konfiguriert die Rollen für die Nutzer in der Umgebung ein.

Anwendungsentwicklersicht

Anwendungsentwickler bezeichnet hier die Zuständigkeit für die Programmierung der Middleware und der Geräte auf denen die Middleware läuft. Der Anwendungsentwickler entwickelt die Middlewaresoftware und stellt sicher, dass die Software einwandfrei auf den Geräten intalliert wird und lauffähig ist. Werden neue Hardware-Schnittstellen entwickelt, dann müssen diese in die Middleware integriert werden. Er definiert die Schnittstellen des Gerätes und legt dadurch fest, mit welchen anderen Gerät es kooperieren kann.

Der Nutzer und Geräteentwickler erwarten von der Middleware, dass die Anwendung einfach arbeitet und abwärtskompatibel ist. Es gibt viele verschiedene Standards (offene, defakto, propritäre), welche berücksichtigt werden müssen, um das Hauptziel, die Interoperabilität, zu gewährleisten.

Gerätehersteller

Der Gerätehersteller entwickelt das Gerät so, dass die Middleware darauf läuft. Die Funktionalität des Gerätes wird semantisch beschrieben, damit das Gerät dynamisch integriert

werden kann. Anhand der semantischen Beschreibungen erfolgt die Integration des Gerätes in die Ontologien. Wie Geräte miteinander kooperieren wird vom Gerätehersteller vordefiniert. So legt dieser fest, das ein Videorekorder ein bestimmtes Videoformat abspielen und aufnehmen kann. Für dem Videorekorder bedeutet, das er mit allen Videoanzeigegeräten kooperieren kann.

Der Gerätehersteller erhält eine Testplattform, um so das Zusammenspiel verschiedener Geräte zu testen.

4.2.2 Residential Gateway

Die logische Komponente Residential Gateway ist für die Anbindung des Systems an externe Netzwerke (zum Beispiel um Web Services aufzurufen) zuständig. Für die Anbindung an externe Diensteanbieter muss diese Netzwerkschnittstelle geschaffen werden. Das Residential Gateway ist die einzige Komponente mit dem Zugriff auf externe Netzwerke. Die einzelnen Geräte, die als Dienste angemeldet sind, stellen eine Anfrage an den HomeController wenn sie Zugriff auf Web Services benötigen und der HomeController koordiniert diese Anfragen und leitet sie an das Residential Gateway weiter. Ein Gerät enthält in seinen Gerätedaten eine semantische Beschreibung des Geräteherstellers. Bei Anfragen und Problemen wird die Hersteller-URL aufgerufen und eine Problembehandlungsstragie heruntergeladen. Falls keine Problemlösung durch den Nutzer erfolgen kann und der Nutzer eine Terminkalenderanwendung besitzt, dann kann diese Termine mit einem externen Dienstleister ausmachen. Beliebige Termine (Arzttermine, Reisetermine etc.) können vereinbart werden, wenn die Anbieter die entsprechenden Schnittstellen unterstützen und diese semantisch beschrieben haben. Desweiteren können über das Gateway aktuelle Reise- und Wetterdaten abgerufen werden. Die Kommunikation mit den externen Diensten erfolgt über Semantische Web Services [8]. Der Zugriff auf den HomeController per Webinterface wird über das Residential Gateway realisiert. Der Nutzer muss in seinen Policen genau festlegen, welche Dienste externe Web Services aufrufen und Daten nach außen schicken dürfen. Die einzelnen Anwendungen müssen konfigurierbar sein, damit anwendungsspezifische Policen einstellbar sind.

Der Nutzer ist im Fehlerfall eines Gerätes daran interessiert, dass eine Lösung zur Behebung des Fehlers schnell gefunden wird. Dazu muss mit dem Hersteller kommuniziert werden und ein Reparaturtermin vereinbart werden. Aus Datenschutzgründen sollte, jedoch bei diesem Vorgang, der Hersteller nicht über alle freien Termine des Nutzers informiert werden. Damit die bestmögliche Sicherheit und ein hohes Maß an Privatsphäre gewährleistet werden kann, werden Regeln anwendungsbezogen definiert, beziehungsweise für einen konkreten Anwendungsfall, der von unterschiedlichen Anwendungen implementiert werden kann.

4.2.3 Knowledge Base

Die Knowledge Base (KB)ist der zentrale Aufbewahrungsort für Ereignisse, Ontologien und Regeln. Folgende Fakten sind in der Knowledge Base gespeichert.

- Sämtliche Ereignisse, die auftreten können, werden in die Domänenontologie übernommen.

- Die Ontologien beschreiben die Anwendungsdomäne.
- Regeln für die Auswertung durch die Rule Engine.
- Policies
- Geräte (Haushaltsgeräte, Unterhaltungsgeräte), Steuerungsgeräte (Heizung, Licht), Sensoren
- Nutzerdaten und deren Präferenzen
- Definierte Situationen in der Wohnumgebung
- Tasks und Aktionen

Ein Administrator gibt in die Knowledge Base neue Ontologien ein, erweitert bestehende Ontologien um neue Konzepte und modifiert Regeln.

Auf die Knowledge Base wird von unterschiedlichen Komponenten zugegriffen, die auf diese Fakten zugreifen um diese abzurufen beziehungsweise neue Fakten hinzufügen.

Die Komponente Police Interface speichert Policen direkt in der Knowledge Base ab, oder diese werden vom Police-to-Rule Transformer in Regeln transformiert und diese werden abgespeichert.

Die Rule Engine benötigt Zugriff zum Ausführen der Regeln. Die Device Registry speichert die registrierten Geräte in der KB ab und der Classifier benötigt Wissen aus der KB um die Geräte korrekt zu klassifizieren.

4.2.4 Device Cooperation Manager

Der Device Cooperation Manager ist für die Verwaltung der tatsächlichen Kooperation der Geräte verantwortlich. Er wird von der Rule Engine über ausgelöste Regeln informiert, verarbeitet diese Informationen und steuert Geräte durch Aufrufen der entsprechenden Dienste auf den Geräten.

Ein Gerät bietet einen Dienst an und dieser soll anhand eines Regelwerkes dynamisch aufgerufen werden. Wie ist der Weg, von einem Ereignis, welches auftritt bis zum Punkt, dass ein Dienst aufgerufen wird? Zuerst tritt ein Ereignis auf, dieses wird verarbeitet und zu einer Aktion, dann entsteht daraus ein Task und dieser wird auf einen Dienst abgebildet. In der Rule Engine werden Regeln ausgelöst, wenn bestimmte Bedingungen von Regeln erfüllt sind. Durch eine Regel, die auslöst entsteht ein Ergebnis, welches wiederum Eingabe-Bedingung für die nächste Regel sein kann. So wird eine Regelkette ausgeführt. Zuerst wird ein geeignetes Gerät gefunden, dann muss der Status passen und der Kontext. Wenn all diese Bedingungen erfüllt sind, feuern die entsprechenden Regeln. Am Ende dieser Regelkette gibt es eine Regel, die eine Aktion erzeugt. Aus einer Aktion wird ein Task abgeleitet. Dieser Task wird direkt vom Device Cooperation Manager auf einen Dienst abgebildet, der von einem Gerät oder mehreren Geräten ausgeführt wird. Der Device Cooperation Manager kümmert sich somit um die Abbildung von Tasks auf phyische Geräte und ruft deren Dienste mit den entsprechenden Parametern auf.

Zwischen den unterschiedlichen Diensten kann es zu Abhängigkeiten kommen, die Probleme

in der Ausführung verursachen können. Der Device Cooperation Manager erkennt diese Abhängigkeiten und bringt die Ausführung der Tasks in die richtige Reihenfolge. Dazu ermittelt er welche Geräte für die Ausführung des Tasks, also für eine Kooperation, zur Verfügung stehen. Die benötigten Daten erhält er aus der Knowledge Base und aus der Device Registry. Treten Änderungen in diesen Quellen auf, wird er mit Ereignissen darüber informiert.

Müssen mehrere Dienste kombiniert werden, um einen Task auszuführen, dann übernimmt der Device Cooperation Manager die Service Komposition und Orchestrierung.

4.2.5 Multimodaler Benutzerschnittstellen Manager

Diese Komponente ist für die Anpassung von Eingaben und Ausgaben, an die geeigneteste Ein- und Ausgabemodalität, verantwortlich. Die Benutzerschnittstelle wird abstrakt definiert und kann so dynamisch, an die unterschiedlichsten Geräte und Anwendungen, angepasst werden. Das Ziel ist eine optimierte Benutzerschnittstelle, die dynamisch an die Nutzerpräferenzen, das jeweilige Endgerät, dessen Darstellung und Performance angepasst wird. Diese Komponente soll festlegen, wie die Funktionalität von verschiedenen Geräten, am Besten auf einem Gerät integriert und dargestellt wird. Nachdem das System ein Gerät, für die Interaktion mit dem Nutzer, ausgewählt hat, welches die Aufmerksamkeit des Nutzers hat, erfolgt die Adaption der abstrakten Benutzerschnittstelle an das ausgewählte Gerät.

Diese Komponente arbeitet mit der Nutzermodellierungskomponente und der Context Service Komponente zusammen, da bei der Adaption von Modalitäten immer die Nutzereigenschaften und Nutzerpräferenzen berücksichtigt werden.

Die Benutzerschnittstelle passt sich an den Kontext des Nutzers an, dass heißt an seinen Ort, das Interaktionsgerät oder seine Behinderungen [6]. Diese Komponente nutzt Kontextinformationen, um die Abpassung an den Nutzer vorzunehmen. In Abhängigkeit von der Tätigkeit des Nutzers und seiner Situation, wird die Ausgabemodalität und die Eingabemodalität angepasst. Die Eingabe des Nutzer kann per Tastatur, Maus, Stift, Spracheingabe oder irgendeine andere Eingabeschnittstelle erfolgen. Die Nutzung diverser Eingabemethoden wird durch die aktuelle Aktivität des Nutzers eingeschränkt. Ein Nutzer der gerade kocht, kann nicht oder nur schwer per Tastatur, Maus oder per Stift mit einem Gerät interagieren. In diesem Fall, ist die Spracheingabe und Sprachausgabe für die Interaktion mit einem Gerät besser geeignet.

In [28] wird ein Ansatz beschrieben wie die Modellierung multimodaler, adaptiver Benutzerschnittstellen von Werkzeugen unterstützt werden kann. Mit diesem Ansatz lassen sich die geforderten abstrakten Benutzerschnittstellen erzeugen.

Eine Kernfrage des Benutzerschnittstellen-Designs ist es, die Benutzerschnittstelle so zu gestalten, dass der Nutzer diese leicht handhaben kann und hier insbesondere der ältere Nutzer, die Funktionen und Aufgaben eines jeden dargestellten Dienstes versteht.

Abstrakte Benutzerschnittstelle

Um die Anforderung *Kontrollieren von Funktionen eines entfernten Gerätes von einem geeignetem Gerät aus (Fernbedienung)* umzusetzen, müssen Funktionalitäten eines entfernten Gerätes auf dem lokalen Gerät dargestellt werden. Die Funktionalitäten müssen abstrakt definiert werden, so dass sie auf den verschiedenen Gerätentypen angezeigt werden können und der Nutzer dieses Funktionen aufrufen kann. Eine Fragestellung die dabei aufgeworfen wurde ist und in weiteren Arbeiten näher zu untersuchen ist es herauszufinden wie die Funktionalität eines entfernten Gerätes auf dem aktuellen Gerät des Nutzers integriert und präsentiert werden kann. Wenn Daten eingegeben werden und Kontrollelemente betätigt werden, dann müssen diese erfasst und zur Ausführung an das entfernte Gerät übermittelt werden. Untersucht werden muss, inwieweit und wie das bei existierenden Lösungen funktioniert. Parameter und Rückgabewerte eines Dienstes werden mit in die Geräteontologie aufgenommen.

Diese Komponente wird von den anderen Komponenten als kontextabhängiger Dienst genutzt.

4.2.6 User Modeling & User Profiling

In dieser Komponente wird der Nutzer und seine Präferenzen modelliert. In einem Nutzermodell werden die Daten zu einem Nutzer abgespeichert. Diese wird in der Knowledge Base verwaltet. Damit das System den Nutzer bestmöglich unterstützen kann, muss das System Kenntnisse über Eigenschaften, Präferenzen und Gewohnheiten des Nutzers haben. Dazu gehört es unter Anderem die Interaktionsmodalitäten (Eingabe- und Ausgabemodalitäten, mit denen der Nutzer mit einem System interagieren kann) des Nutzers zu berücksichtigen. Diese hängen vom jeweiligen Nutzer ab und können, im Vergleich zu einem angenommenen sterotypischen Normalnutzer, eingeschränkt sein. So hat ein Nutzer, der nicht sehen kann, keine Möglichkeit visuelle Inhalte wahrzunehmen und kann nicht mit grafischen Eingabegeräten interagieren.

Es gibt vorgefertigte Modelle von stereotypischen Nutzern, aus denen das Modell ausgesucht wird, welches dem Nutzer am Ähnlichsten ist. Des Weiteren gibt es vordefinierte Benutzerrollen wie Bewohner, Familie, Freund, Nachbar, Pflegepersonal usw. Dazu werden dem Nutzer, bei erstmaliger Initialisierung des Systems, Fragen zu seinem Verhalten und seinen Präferenzen gestellt. Eine andere Variante zur Erfassung von Nutzerdaten für die Bildung eines Profils ist eine Lernphase, in der der Nutzer beobachtet wird. Basierend auf dem gelernten Nutzerprofil, wird dem Nutzer das ähnlichste sterotypische Modell vorgeschlagen. Damit wird der Aufwand zur Erstellung von Profilen für den Nutzer reduziert, da das System selbständig lernt und der Nutzer nur vorgefertigten Profilen an seine Bedürfnisse anpassen muss.

Der Nutzer muss jederzeit auf die Nutzermodellierung vollen Zugriff haben und den Prozess der Modellierung abschalten können [16].

4.2.7 Policy Interface

Die Komponente Policy Interface ermöglicht es dem Nutzer, über die Benutzerschnittstelle, Policen in das System einzugeben. Der Nutzer hat das gleiche Interface unabhängig vom Hersteller des Gerätes. Die Policies werden unabhängig von Regeln spezifiziert.

4.2.8 Policy Manager

Der Policy-Manager verwaltet die Policen, also Richtlinien (Regeln auf hohem Niveau) mit denen das low-level Systemverhalten kontrolliert und angepasst wird. High-Level Regeln betreffen Aspekte der Privacy und Präferenzen. Eine Beschreibung von Policen ist in Unterabschnitt 4.1.1 zu finden.

4.2.9 Policy-to-Rule Translator

Diese Komponente übersetzt Policies aus der obersten Schicht der dreischichtigen Kooperationsarchitektur (siehe Abbildung 4.1) in niedrigschichtigere Regeln.

4.2.10 Regelschicht

Diese Schicht enthält die eigentliche Logik dieses Systems. In einem Repository werden alle Regeln gespeichert, die vordefiniert sind und die vom Nutzer zusätzlich definiert werden. Auf Basis von diesen Regeln wird die Kooperation und Integration der beteiligten Geräte festgelegt. Die Regelschicht erhält die aktuellen Kontextdaten aus der Kontextschicht und feuert bei Kontextänderungen die entsprechenden Regeln, die daraufhin ausgeführt werden. In Unterabschnitt 4.1.2 wird beschrieben, wie solche Regeln aufgebaut sind.

4.2.10.1 Rule Engine

Die Rule Engine greift auf die in der Knowledge Base vorhandenen Regeln und die dazugehörigen Kontextdaten und Ereignisse zu und wertet die Regeln damit aus.

4.2.10.2 Regel Monitor

Der Rule Monitor wird von der Kontextschicht über Kontextereignisse informiert. Er überprüft alle Regeln auf einen passenden Event-Teil, welcher der aktuellen Situation entspricht. Es können mehrere Regeln gefunden werden, deren Event-Teile übereinstimmen und dadurch können Konflikte auftreten, wenn diese gleichzeitig gefunden werden oder wenn Dienste, die im Actions-Teil ausgeführt werden sollen, exklusiv sind, also z.B. dieselben Ressourcen beanspruchen. Diese Konfliktregeln werden an eine andere Komponente, den Regel Konfliktmanager weitergeleitet und von diesem aufgelöst, bevor die entsprechenden Dienste aufgerufen werden.

4.2.10.3 Regel Konfliktmanager

Wenn Konflikte zwischen verschiedenen auslösenden Regeln auftreten, dann müssen diese aufgelöst werden. So kann es sein, dass im Aktionsteil einer Regel ein Gerät verwenden soll, welches dann aber die Ausführung der anderen Regeln blockiert, oder nur einen exklusiven Zugriff hat. Soll beispielsweise ein Videorekorder von einer Videoquelle aufzeichnen, dann kann das nur von einer Quelle zur gleichen Zeit erfolgen. Das sind gerätenahe Regeln, die von dem jeweiligen Gerät an das System übermittelt werden müssen und die sich aus der Spezifikation des Dienstes ergeben.

Dazu haben die einzelnen Regeln eine Priorität, die ausgewertet wird. Diese ist abhängig von der Aktion, die durch diese Regel ausgelöst werden soll. Des Weiteren hängt die Priorität einer Regel von der Rolle des Nutzers ab, für den die Regel gefeuert wurde. Aus der Rolle des Nutzers ergibt sich, in welcher Reihenfolge Regeln ausgeführt werden.

4.2.10.4 Regel Entdecker

Diese Komponente findet Muster im Nutzerverhalten mithilfe der *frequent pattern discovery* [13]. Wenn der Nutzer mehrfach dieselbe Aktion in einer bestimmten Situation ausführt, dann wird das vom System erkannt und es generiert neue Regeln. Regeln werden entweder vom System gelernt oder direkt vom Nutzer oder Administrator eingeben. Aus Nutzerentscheidungen und dem Nutzerverhalten werden Präferenzen abgeleitet, die bei zukünftigen Entscheidungen berücksichtigt werden. Dazu wird der aktuelle Kontext berücksichtigt. Eine alternative Herangehensweise für die Umsetzung des Rule Discoverer, ist der Decision-Learner, den Patrick Wustmann in seiner Diplomarbeit [29] entworfen hat. Dieser lernt aus Nutzerentscheidungen und bestimmten Kontextdaten. Daraus erstellt er eine Regelbasis und wenn wieder eine Situation mit diesem Kontext eintritt, dann entscheidet das System für den Nutzer.

4.2.10.5 Reasoner

Ein Reasoner vergleicht die aktuellen Objektinstanzen mit den vorhanden Regeln in der Regelbasis und aktiviert eine Regel, wenn die Bedingungen der Regeln erfüllt sind. Dazu muss der Reasoner auf die Regelbasis und die Knowledge Base zugreifen.

4.2.10.6 Regelbasis

Die Regelbasis enthält alle Regeln, die das Anwendungsverhalten des Systems in Situationen definieren und die Kooperation zwischen Geräten definieren. Dabei wird unterschieden, in systemgenerierte Regeln und in nutzergenerierte Regeln. Aus diesen Regeln werden die spezifischen Entscheidungen für die Kooperation von Geräten, die im Smarthome auftreten können, festgelegt.

4.2.10.7 Regelübersetzer

Die hier beschriebenen Regeln werden in einer formalen Sprache beschrieben. Dem Nutzer werden die Regeln in einer leicht verständlichen Sprache dargestellt, da nicht erwartet werden kann, dass er eine Regelsprache erlernt oder mit einem Ontologie-Editor hantiert. Es muss eine Übersetzung stattfinden zwischen natürlichsprachigen Regeln, die der Nutzer versteht und leicht zu definieren sind oder auswählbar sind, in Regeln die vom System verstanden werden. Ebenfalls müssen Geräte, die mit einer Ontologiesprache beschrieben werden für den Nutzer in eine verständliche Form gebracht werden. Der Nutzer muss die Funktionalität eines bestimmten Gerätes leicht ablesen können.

Dem Nutzer soll verdeutlicht werden, welche Geräte sind vorhanden und welche Aufgaben ergeben sich daraus. Die gesamte Situation des Nutzers, mit seinem Kontext, wird dem Nutzer verständlich dargestellt, so dass er Regeln dafür definieren kann.

4.2.10.8 Regelersteller

Mit einem Regelersteller (Rule Builder) erzeugt ein Nutzer Regeln. Regelsprachen sind nicht einfach zu verstehen und aus diesem Grund, muss es für den Nutzer eine einfache Benutzeroberfläche geben, mit der Regeln erstellt werden können. In [30] werden zwei Prototypen eines Rule Builder beschrieben. Der erste Prototyp bietet eine graphische Repräsentation des JESS Regelkonzeptes. Der zweite Prototype basiert auf einem Beispiel-basierten Ansatz, der Puzzle-ähnliche Teile verwendet. Der Nutzer kopiert einfach existierende Regeln, die vom Entwickler vordefiniert wurden oder teilweise vom RuleDiscoverer empfohlen wurden, und passt diese an.

Jeder Nutzer hat ein sehr unterschiedliches Verhaltensmuster, und daher wird vom Nutzer erwartet, dass er das Systemverhalten durch Hinzufügen oder Entfernen von Bedingungen und Aktionen modifiziert. Beide Prototypen wurden von einer Gruppe von Informatikstudenten evaluiert und dabei wurde ermittelt, dass die Nutzerkontrolle über das Systemverhalten zunimmt, wenn eigene Regeln definiert werden und so das Systemverhalten zur Laufzeit verändert wird. Der erste Prototype, bei dem die Regeln, von Grund auf, in der Regelsprache definiert werden, ist in der Praxis zu kompliziert. Hingegen können beim zweiten Prototypen dieselben Regeln mit erheblich niedrigerem Zeitaufwand, weniger Fehlern und weniger Hilfestellung erstellt werden. In Anbetracht der Tatsache, dass es sich bei den Probanden in diesem Test um Informatikstudenten gehandelt hat, die normalerweise detaillierte Vorkenntnisse, im Umgang mit technischen Geräten, besitzen und dennoch Probleme beim Erstellen der Regeln hatten, zeigt wie wichtig der Aspekt des Regelerstellens für den Einsatz in einem Smarthome für ältere Menschen ist. Der Rule Builder muss eine sehr einfache und intuitive Benutzerschnittstelle haben, die es Menschen mit wenigen technischen Vorkenntnissen ermöglicht, die Smarthome-Anwendung an ihre Bedürfnisse anzupassen.

4.2.10.9 Regel Inspektor

Mit dem Rule Inspektor kann der Nutzer bereits ausgeführte Regeln inspizieren. Der Nutzer kann Regeln nicht nur lesen, sondern auch modifizieren, löschen, aktivieren oder deakti-

vieren. So kann der Nutzer Regeln personalisieren und seinen Bedürfnissen anpassen und bestimmen, wie sich das System in einer Situation verhalten sollte.

Alle ausgeführten Regeln stehen in einer chronologischen Liste. Falls der Nutzer Probleme haben sollte das Systemverhalten zu verstehen, dann kann er die Regeln so nachvollziehen. Je mehr Vorgänge und Situationen vom System automatisiert werden und dem Nutzer auf diese Weise Arbeit abgenommen wird, desto größer ist das Risiko, dass der Nutzer die Übersicht verliert, welche Regeln im Hintergrund ablaufen. Daher ist es wichtig, dass der Nutzer jederzeit Zugriff auf die Ablauf der Regeln informieren kann. Dazu gehören auch verständliche Fehlerberichte, die deutlich machen, warum Regeln nicht ausgeführt werden konnten.

4.2.10.10 Regel Debugger

Diese Komponente ermöglicht es dem Administrator, Regeln zu debuggen. Damit lassen sich ausgeführte Regeln gegebenenfalls rückgängig machen, wiederholen oder können modifiziert werden. Von dem Nutzer kann nicht erwartet werden, dass er das System auf dieser Ebene versteht.

4.2.11 Taskmodellierung

Wenn eine Regel feuert, dann wird aus der Aktion ein Task modelliert. Greift auf Daten in den Ontologien zu, um zu ermitteln, ob Task möglich ist. Falls ein Gerät für die Ausführung eines Tasks benötigt wird, dann kann es aktiviert werden. Der Taskmodellierer greift auf das Repository zu. Die A-Box zu der Geräteontologie wird mit den Geräten instantiiert. Für die Ausführung von Regeln wird auf diese Instanzen zugegriffen.

Task-Modell mit Interaktions-Tasks (Nutzer) und System-Tasks (Systemlogik)

Komplexe Tasks werden aus atomaren Tasks zusammengesetzt.

Aus den vorhandenen Geräten und ihren Fähigkeiten (Eingabe, Ausgabe, Verarbeitung), Kontrollfunktionen und verfügbaren Inhalten werden automatisch Tasks erstellt und dem Nutzer angeboten, der sich dann entscheiden kann, ob diese ausgeführt werden. Es werden dazu Nutzerwünsche, Kontrollanfragen und die aktuelle Situation berücksichtigt. Das beschreibt die Tasks, die sich aus der Konfiguration des Systems und dem aktuellen Zustand ergeben. Tasks können ebenfalls durch Regeln ausgelöst werden, deren Bedingungen erfüllt sind. Dann ist ein Task, die Aktion, die ausgeführt wird und die Logik enthält.

Eine *Task Definition* ergibt sich aus der Schnittstellen der beteiligten Geräte, dem Kontext und wie diese miteinander interagieren können. Aus der Task Definition ergibt sich wie die Geräte miteinander integriert werden. Dazu muss bekannt sein welche Geräte zur Verfügung stehen und wie diese mit einander kommunizieren. Dafür werden im Vorfeld die entscheidenden Geräteparameter genau definiert, Eingabe-/Ausgabe, mögliche Interaktion.

Beispiel ist die Integration von einem Handy mit einem Navigationsgerät. Das Navigationsgerät hat ein größeres Display und auf diesem können SMS besser gelesen und eingegeben werden. Zusätzlich läßt sich ein Telefonat über das Navigationsgerät annehmen, weil dieses

ein Mikrofon und einen Lautsprecher hat und so freigesprochen werden kann. Die Bedienung ist bei hohen Geschwindigkeiten einfacher als auf einem Handy mit kleineren Tasten. Es kann das Telefonbuch an das Navigationsgerät übermittelt werden und so können Adressen aus dem Handy in die Zielführung übernommen werden.

Die Verwaltung der Tasks erfolgt in einem Taskpool, in dem alle angefallen Tasks gespeichert werden und dann in der Reihenfolge ausgeführt werden. Wie diese ausgeführt werden, hängt von der Prioriät eines Tasks ab und welche Ressourcen dieser zur Ausführung benötigt.

Können Tasks immer generiert werden, oder erst wenn die benötigten Geräte und der Kontext zur Verfügung stehen?

4.2.12 Ereignis Manager

Prinzipiell erzeugt ersteinmal alles ein Ereignis. Jedes Gerät, welches an- oder abgemeldet wird ist ein Ereignis. Jede Kontextänderung, ist ein Ereignis, welche vom Kontextdienst registriert wird. Die Anforderungen an den Ereignis Manager sind die Behandlung von Geräteereignissen (Warnung, Fehler, Status, Notfall) und Ereignisse wie Erinnerungen, die den Nutzer direkt betreffen. Aufgabe des Ereignis Managers ist es den Ereignistyp, anhand von Ereignisquelle und Kontext, zu ermitteln. Dazu kommuniziert diese Komponente mit dem Rule Layer und dem Kontextdienst.

Ermittlung von Geräten, die Ereignis darstellen können. Die optimale Darstellung des Ereignis übernimmt dann die Komponente Multimodaler Benutzerschnittstellen Manager.

Ereignisauslösung

Ereignisse können von Sensoren, Geräten, Personen und durch Kontextänderungen (siehe ContextProvider) ausgelöst werden. Damit treten diese Entitäten als mögliche Ereignisauslöser oder Ereignisquellen auf.

Ereignisbenachrichtigung

Wenn ein Gerät am System angemeldet ist, dann werden für dieses Gerät EventListener registriert. Alle Ereignisse, die auf dem Gerät ausgelöst werden, werden an den zentralen Ereignis Manager übermittelt. Dieser leitet das Ereignis an den Rule Layer weiter. Dort wird dann mit Regeln, ermittelt, wie auf das Ereignis reagiert wird.

Ereignisbehandlung

1. Ereignis wird von einem Gerät ausgelöst. Jedes Ereignis wird als Kontextinformation betrachtet.
2. Senden von Ereignis an die Middleware (HomeController). Der Ereignis Manager hat Listener für dieses Ereignis registriert in Middleware und erhält das Ereignis.
3. Ereignis Manager leitet das Ereignis an den Rule Layer weiter.
4. Es feuert eine entsprechende Regel und eine Aktion wird ausgelöst.

5. Aus Aktion wird ein Task erzeugt, der ausgeführt wird.

6. Finden des passenden Empfängers und Ausführen des Tasks.

Ereignisklassen

Ereignisse werden in Ereignisklassen unterteilt und semantisch beschrieben. Das Format des Ereignisses ist standardisiert, so dass es von der Middleware weiterverarbeitet werden kann. Durch die Regeln, welche in der Knowledge Base definiert sind, wird die Reaktion auf ein spezifisches Ereignis bestimmt (Ausgabeort und -art).

Ereignisse klassifiziert nach Ereignistyp und Priorität:

1. Status

2. Warnung

3. Fehler

4. Notfall

Diese Klassifizierung stellt Oberklassen dar und kann weiter differenziert werden, in verschiedene Notfälle. Lässt sich für jedes Ereignis eine Standardverhalten in einer Situation festlegen? Der Gerätehersteller ordnet jedes Ereignis, dass auf dem Gerät auftreten kann in die hier entwickelte allgemeine Ereignistypkategorie ein. Aus der Kategorie ergibt sich das Standardverhalten (Reaktion), welches bei Auftreten des Ereignisses angewendet werden soll. So soll beispielsweise bei einem Ereignis, der Kategorie Notfall, die Feuerwehr alarmiert werden. Es soll eine weitere Ontologie geben, die das Wissen enthält, wer benachrichtigt werden soll bei welchem Ereignistypen.

Ereignis und Aktion werden durch die Middleware voneinander entkoppelt. Es gibt keine direkte Verknüpfung zwischen Ereignis und Tasks und den Regeln. Im System gibt es vordefinierte Tasks für bekannte Ereignistypen.

Die Ereignisquelle bezeichnet den Ursprung eines Ereignisses. Sozusagen den Ort, an dem das Ereignis ausgelöst wurde. Wenn der Kontextdienst zu dieser Ereignisquelle befragt wird, können weitere interessante Kontextinformationen abgerufen werden. So kann der Standort des Geräte ermittelt werden, wer mit dem Geräte interagiert hat. Aus der Interaktionsgeschichte des Gerätes kann ermittelt werden für welchen Nutzer ein auftretendes Ereignis interessant sein könnte. *Hat die Dame Janet die Waschmaschine eingeschaltet, dann möchte sie in der Regel darüber informiert werden, dass diese fertig ist. Diese Statusinformation ist für die dreijährige Enkelin, die gerade ein Bild malt, und meistens auch für andere Nutzer irrelevant.* Aus diesem Grund ist es wichtig, einen Ereignistypen festzulegen. Aus diesem muss sich ableiten lassen, wie die konkrete Benachrichtigung auszusehen hat. Das bedeutet welche Personen könnten sich für das Ereignis interessieren und wer muss auf jeden Fall benachrichtigt werden und wie. Ereignisse vom Typen Interaktionsereignis, benötigen eine Interaktion des Nutzers. Es muss folglich ein Gerät gefunden werden, auf dem dieses Ereignis angezeigt werden kann und mit dem der Nutzer interagieren kann. Dabei muss noch die Adaptation an das geeignetste Gerät vorgenommen werden, diese Aufgabe wird von dem Multimodaler Benutzerschnittstellen Manager übernommen.

Ohne Verwendung eines Kontextdienstes, könnten einfach alle Geräte eines Nutzers unter seinem Namen identifziert werden und der Nutzer wird nur von Ereignissen benachrichtigt, die auf diesen beteiligten Geräten ausgelöst werden.

Ereignisbehandlung

Geräte, die sich anmelden können sich für Ereignistypen registrieren (Vergleich EventLister), dass entspricht dem herkömmlichen Ereignismechanismus. Bei dem hier gewählten Ansatz, melden sich die Geräte an und dann legt der HomeController fest, für welche Ereignisse dieses registriert wird. Der EventListener muss somit nicht zu Beginn festlegen für welches Ereignis er sich interessiert, sondern das kann dynamisch zur Laufzeit bestimmt werden. Die Abonnierung übernimmt der Controller als Stellvertreter und dieser kennt mögliche Geräte, die ein Ereignis verarbeiten können.

4.2.13 Context Service

Der Kontextdienst (Context Service) hat die Aufgabe Kontextdaten zu erfassen, zu verarbeiten und diese an andere Komponenten, welche sich für Kontextdaten interessieren, zu liefern. Dieser kann relevanten Kontext von den verschiedensten Kontextquellen ermitteln und zusammenfassen. Aus low-level Kontext leitet der Kontextdienst höherwertigen Kontext wie Situationen, Aktivitäten ab. Diese werden auf Anwendungsschicht der Anwendung zur Verfügung gestellt.

Beim Kontextdienst werden alle Geräte mit ihrem Kontext registriert. Er wird über alle Änderungen, die den Kontext betreffen, informiert. Sie erlaubt es Anwendungen, die an Kontextdaten interessiert sind, sich für Kontextänderungen anzumelden. Die Anwendungen werden dann über alle relevanten Änderungen, bezüglich eines bestimmten Kontextattributes oder einem höheren Kontextwert, informiert.

Der Kontextdienst wird von verschiedenen Komponenten genutzt, die Kontext benötigen. Die Rule Engine ist mit dem Kontextdienst verbunden um auf die Kontextdaten zuzugreifen, die benötigt werden. Wenn die Rule Engine Anbindung an die RuleEngine zur Übermittlung sämtlicher benötigter Kontextdaten. Anwendungen registrieren Regeln, die spezifizieren, bei welchen Änderungen des Kontextes sie benachrichtigt werden sollen.

Modelling

Von den erfassten Kontextdaten werden Instanzen zu den Konzepten im Kontextmodell erzeugt. Das ontologiebasierte Kontextmodell enthält eine Beschreibung der Domäne und eine Beschreibung der Gerätefunktionalitäten. Eine Ontologie wird verwendet, weil diese eine bessere Interoperabilität zwischen verschiedenen Anwendungen bietet. Desweiteren hat eine Ontologie eine höhere Aussagefähigkeit als andere Ansätze zur Kontextbeschreibung (siehe Unterabschnitt 3.1.1). Es wird dem Vorschlag von [4, 11] gefolgt und ein mehrschichtiger Ansatz gewählt, mit einer oberen Ontologie und mehreren domänenspezifischen Ontologien. Die domänenspezifische Ontologien beschreiben, in dieser Arbeit, die Wohnumgebung, den Nutzer, die Funktionalität der Geräte und Tasks (siehe Abschnitt 4.3).

Für das hier zu verwendene Kontextmodell werden bereits existierende Ontologien wiederverwendet und erweitert. Hier wird die CONtext Ontology (CONON) und die Home-

Domain ontology wiederverwendet. Diese Ontologien beschreiben alle Aspekte der Wohnumgebungsdomäne. Es werden bereits existierende Ontologien, die die Smarthome-Domäne beschreiben, aus dem AMIGO-Projekt wiederverwendet. Damit wird der Aufwand für die Erstellung der Ontologien reduziert.

Eine intelligente Umgebung muss relevante Informationen über die Vorgänge im Innern sammeln, was als Kontext bezeichnet wird, um kontextbewußtes Verhalten zu zeigen. Besonders relevant ist es die Kontextdaten des Nutzers zu erfassen. Dazu gehören der Aufenthaltsort des Nutzers, seine Aktivität und die Anwesenheit anderer Menschen. Um zu ermitteln mit welchen Geräten ein Nutzer interagieren kann, muss die Position der Geräte bekannt sein und Kontextdaten über deren Status, wer das Gerät momentan bedient, wer sich in der Nähe befindet und weitere sind von Nutzen.

Reasoning

Die Reasoning Komponente schlussfolgert über Kontext und leitet höherwertigen Kontext aus niederwertigen Kontextdaten ab.

Storage

Die Storage Komponente kümmert sich darum die Kontextdaten in einer geeigneten Form abzuspeichern. Es gibt eine Assoziation zur Knowledge Base und dort werden alle Kontextdaten als Individuen in den Ontologien abgespeichert.

Discovery

Die Discovery Komponente ist dafür verantwortlich Kontext zu finden.

EventListener

Der EventListener hört auf Kontextänderungen und Ereignisänderungen und aktualisiert die modellierten Kontextdaten.

4.2.14 Device Registry & Classification

Die Komponente Device Registration &Classification verwaltet alle im System angemeldeten Geräte und klassifiziert sie entsprechend der Gerätefunktionalitäten in die Geräteontologie ein. Fakten die von Geräten relevant sind, werden durch die Geräteontologie bestimmt. In erster Linie sind das die Dienste, die das Gerät zur Verfügung stellt. Weiterhin ist der Kontext des Gerätes relevant, dass heisst wo befindet es sich und wer bedient es.

4.2.14.1 Device Registry

Wenn sich das Gerät in der Anmeldungsphase beim HomeController anmeldet, dann wird das Gerät in die Device Registry eingetragen. Die Dienste eines Geräte werden mit einer semantischen Beschreibungssprache für Dienste beschrieben und bei der Anmeldung an

den HomeController wird diese Beschreibung an die Device Registry übertragen.

Wird ein Gerät aus der Anwendungsdomäne entfernt, dann wird es abgemeldet und aus der Device Registry ausgetragen. Meldet sich ein Gerät nicht in einem festgelegten Zeitintervall, dann verbleibt es in der Device Registry aber der Status wird auf unbekannt geändert. Wird ein Gerät ausgeschaltet, dann verbleibt es im Repository und der Status wird auf inaktiv gesetzt. Das Gerät wird bei der Ereignisbenachrichtigungs-Komponente registriert und somit werden alle Ereignisse vom HomeController registriert und an die dafür registrierten EventListener propagiert.

Der aktueller Zustand des Gerätes wird durch seinen Kontext repräsentiert. Die Device Registry informiert die Knowledge Base über die Änderung und fügt das neue Gerät hinzu. Die Rule Engine wird von der Knowledge Base über Änderungen in der Wissensbasis informiert. Ab diesem Zeitpunkt steht das Gerät für Regeln zur Verfügung und wenn eine Regel ausgelöst wird, dann wird mit dem Gerät ein Task ausgeführt.

4.2.14.2 Device Classification

Die Komponente Device Classification ist für die Modellierung von Geräten zuständig. Anhand der semantischen Beschreibungen der Dienste eines Gerätes erfolgt die Abbildung, in die verschiedenen Kategorien der Geräteontologie, automatisch. Ein Gerät, welches mehrere Dienste anbietet, wird in mehreren Kategorien eingeordnet.

Geräte werden modelliert und in Geräteklassen eingeordnet. Der erste Schritt ist dabei die Geräteschnittstellen abzuleiten. Daraus wird eine Hierarchie von Schnittstellen ermittelt und Funktionalitäten in abstrakte Oberklassen ausgelagert. Der nächste Schritt ist es eine allgemeine Geräteklassifikation in einer Ontologie darzustellen. Geräte werden unterschieden in allgemeine Kategorien wie Unterhaltungsgeräte, Haushaltsgeräte und Steuerungsgeräte. Die Gerätefunktionalität wird abstrakt beschrieben, da sich die verschiedenen Geräte sehr voneinander unterscheiden. Durch die Beschreibung der Geräte in einer Ontologie, lässt sich dieses System programmiersprachen- und plattformunabhängig umsetzen.

Damit die Geräte den Konzepten in den Regeln und Tasks entsprechen, erfolgt eine Abbildung zwischen der Beschreibung eines Gerätes und diesen Konzepten.

Wenn das Gerät dem System schon bekannt ist, dann werden die Einträge - Fähigkeiten, Status, aktuell vom Gerät ausgeführter Task - in der Device Registry mit dem aktuellen Profil abgeglichen. Daraufhin wird das Profil in der Knowledge Base aktualisiert. Der Kontextdienst wird über alle Kontextänderungen eines Gerät informiert, die bei ihm registriert wurden und der Kontextdienst benachrichtigt alle Abonnenten.

Die Discovery und Publishing von Diensten erfolgt mit semantischen Attributen. Die Integration und Komposition von Diensten erfolgt aufgrund der semantischen Beschreibung von Diensten.

4.2.14.3 Semantic Service Discovery

Diese Komponente dient dazu zu vorhandenem Kontext den passenden Dienst zu finden. In einer Ubiquitous Computing Umgebung sind potentiell viele Geräte und damit viele Services vorhanden [23]. Diese Komponente ermittelt zu einer Anfrage mit semantischen Attributen den geeigneten Dienst (siehe dazu Semantic Web Services).

In [26] wird vorgeschlagen relevante Dienste, in Pervasive Environments, mit Hilfe von Semantik und Kontext zu finden. Sie sagen, dass eine auf String-Vergleich basierende Service-Beschreibung und Suche in Pervasive Environments nicht funktionieren kann, da jeder Gerätehersteller und somit Service-Anbieter eine unterschiedliche Beschreibung verwendet. Aus diesem Grund muss die Beschreibung von Diensten semantisch erfolgen. Daher erfolgt in dieser Arbeit die Umsetzung mit OWL und einem kontextabhängigen Ontologiebasierten Mechanismus zum Finden der Dienste.

Aus der Device Registry ermittelt die Komponente die verfügbaren Dienste und wählt aus diesen den passenden Dienst aus. Die Komponente liefert auf die Anfrage den den gefundenen Dienst als Resultat zurück. Im nächsten Schritt wird zu dem Dienst das Gerät gefunden, welches diesen Dienst anbietet.

Diese Komponente dient dazu passende Dienste, nach semantischen Kriterien und in Abhängigkeit des Kontextes, zu finden.

Beispiel: Wenn ein Dienst für die Videoausgabe gesucht wird, dann wird gesucht nach dem Kriterium Videoausgabe mit den Parametern Bildschirmauflösung, Anzahl der Farben und Bildwiederholrate. Bei positiver Suchanfrage wird ein passender Videoausgabe-Dienst zurückgeliefert.

4.2.15 Privatssphären Manager

Der Privacy Manager regelt die Aspekte der Privatssphäre, die in einer Smarthome Umgebung sehr wichtig sind. Nutzer wollen entscheiden, wer Zugriff auf ihre Daten hat und möchten nicht, dass jeder Gast in ihrem Haus Einstellungen verändern kann. Besonders relevant ist dies, weil dieses System den Nutzerkontext überwacht und so jederzeit bekannt ist, wo sich der Nutzer aufhält und was er macht. Die Privatsphäre ist das höchste Gut des Einzelnen und es muss gewährleistet sein, dass sie nicht für Aspekte der Bequemlichkeit blossgestellt wird.

4.2.16 Netzwerkabstraktionsschicht

Die Netzwerkabstraktionsschicht abstrahiert die unterschiedlichen Netzwerktechnologogien und stellt der Anwendung eine einheitliche Netzwerkschicht zur Verfügung. Geräte, die sich mit der Middleware verbinden wollen, können dies über eine unterstützte Netzwerkschnittstelle tun. Auf die verschiedenen Heimvernetzungsstandards wie UPnP, HAVI, Bluetooth, Wlan oder andere Technologien zur Verbindung der Geräte auf Netzwerkschicht soll in dieser Arbeit nicht weiter eingegangen werden. Es wird angenommen, dass die Geräte auf Netzwerkschicht miteinander verbunden sind, und die hier beschriebene Middleware ver-

bindet die beteiligten Geräte in der Anwendungsschicht des ISO/OSI-Schichten-Modells.

4.2.17 Plattform

Das System läuft auf allen verbreiteten heterogenen Hardware-Plattformen wie PCs, Eingebetteten Systemen, Mobilen Geräten, Unterhaltungselektronik-Geräten. Die Middleware läuft auf einer Virtuellen Maschine und ist so unabhängig von einem bestimmten Betriebssystem.

4.3 Ontologien

In diesem Abschnitt wird die Verwendung von Ontologien in der entworfenen Architektur beschrieben. In den Ontologien werden die Konzepte des Smarthomes beschrieben, dass heisst es wird der Kontext modelliert, der zur Ausführung der Smarthome-Anwendung benötigt wird. Dazu gehört die Beschreibung der übergeordneten Domänen-Ontologie. Diese enhält die spezifischen Sub-Ontologien zur Beschreibung von Geräten, den Nutzern und von Tasks.

Ein Gerät meldet sich beim Home Controller an und übermittelt Beschreibungsdaten, zu seinen Funktionalitäten an das Repository. Die Ontologie wird dann mit folgenden Daten initialisiert:

- Standort des Gerätes in der Anwendungsdomäne (hier: Wohnung oder im Haus). Dieser Kontext wird benötigt für die Kooperation.
- Nutzer, die mit dem Gerät interagieren.
- Daten, die den Nutzer beschreiben.
- Ereignisse, die vom Gerät ausgelöst werden können.
- Gerätespezifische Policen, welche festlegen wer das Gerät bedienen darf und welche Funktionalität des Gerätes aufgerufen werden dürfen.

Während der Laufzeit werden vom Gerät aktuelle Statusdaten übertragen, wie der aktuelle Zustand, Ereignisse und die Daten in der Ontologie werden damit aktualisiert.

4.3.1 Domänen-Ontologie

Die Domänen-Ontologie repräsentiert hier den Anwendungskontext der Smarthome-Domäne und enthält alle Konzepte, die in der Hausumgebung auftreten können. Die Domänen-Ontologie wird erweitert um eine Geräte-Ontologie zur Beschreibung der Geräte-Konzepte, um die Nutzer-Ontologie, welche die Nutzerkonzepte beschreibt und die Task-Ontologie. Die Ontologie ist um beliebige neue Konzepte erweiterbar durch Hinzufügen der neuen Subdomänen.

4.3.2 Geräteontologie

Die Geräteontologie beschreibt die Konzepte der Geräte, die sich in dieser Domäne befinden. Es werden die Schnittstellen des Gerätes im Detail beschrieben. Das sind Beschreibungen der Funktionalitäten, mit welchen Geräten eine Kooperation erfolgen kann, welche Datentypen werden unterstützt und welche Regeln und Policies werden von einem Gerät unterstützt.

4.3.3 Nutzerontologie

Die Nutzerontologie enthält Wissen über Nutzerpräferenzen, über den Aufenthaltsort des Nutzers und seine Aktivität. In dieser Ontologien werden auch Beziehungen, zwischen verschiedenen Nutzern abgebildet. Daraus lassen sich Rückschlüsse für eine bestimmte Bedienung des Systems ziehen.

Einem Nutzerkonzept ist eine Aktivität zugeordnet und eine Situation, in der sich der Nutzer befindet. Dieses Nutzerkonzept wird mit aktuellen Kontextdaten gefüllt. Dem Nutzerkonzept sind weiterhin zugeordnet, welches Geräte oder welche Geräte vom Nutzer verwendet werden. Aus der Aktivität und der Nutzung eines Gerätes, läßt sich auf die Situation des Nutzers schliessen. Die Aktivität muss berücksichtigt werden, wenn der Nutzer mit einem Gerät interagiert beziehungsweise Geräte kooperieren, damit der Nutzer einen Task ausführen kann. Enthält die Beschreibung der Geräte, mit deren Funktionalitäten wie Eingabe-und Ausgabemodalitäten.

4.3.4 Taskontologie

In der Taskontologie werden die Konzepte von Tasks beschrieben. Für jeden Task ist definiert, welche Gerätefunktionalität zur Ausführung benötigt wird. Das heißt in dem Task steht, welche Dienste dieser verwendet, und mit welchen Parametern. Die zur Laufzeit verfügbaren Tasks ergeben sich aus der Gerätekonfiguration und die Anwendung greift darauf zu, um dem Nutzer mögliche Tasks anzubieten.

4.4 Entscheidungsbäume

Entscheidungsbäume (Decision trees), werden verwendet um komplexe Entscheidungsabfolgen zu modellieren. In Abbildung 4.3 ist ein Entscheidungsbaum für das Ereignis, dass die Waschmachine nicht abpumpt, abgebildet.

Abbildung 4.3: Beispiel eines Entscheidungsbaums

4.5 Zusammenfassung

Dieses Kapitel hat die Konzeption der Architektur vorgestellt. Die prototypische Umsetzung dieser Architektur mit den benötigten Komponenten wird in Kapitel 5 beschrieben. Es wurde desweiteren vorgestellt wie die Middleware Regeln und Policen einsetzt, um damit die Koooperation und Integration von Geräten zu ermöglichen. Entscheidungsbäume werden kurz gezeigt als Methode um komplizierte Entscheidungen, die aus vielen Einzelentscheidungen bestehen, zu modellieren.

5 Implementierung

Dieses Kapitel stellt den Prototypen vor, der während dieser Diplomarbeit entwickelt wurde. Der Prototyp soll die Machbarkeit einzelner Konzepte demonstrieren. Desweiteren hilft der Prototyp dabei, sich über benötigte Funktionalitäten des Systems bewusst zu werden. Es ist ein vertikaler Prototyp (ein vertikaler Schnitt durch das System) der alle Funktionalitäten von der obersten Schicht bis zur untersten Schicht berücksichtigt.

Der Prototyp setzt das Szenario *Anzeige einer Fehlermeldung auf einem beliebigen Gerät, welches die Aufmerksamkeit von Janet hat* aus dem Abschnitt 2.1 um. Die daran beteiligten Komponenten und Geräte werden als OSGi-Bundles umgesetzt. Das sind der Home Controller, ein Videorekorder, ein Fernseher und ein Telefon. Jedes dieser Geräte wird auf seine Wesentlichen hier benötigten Funktionalitäten reduziert. Für jedes Gerät wurde eine grafische Benutzerschnittstelle programmiert, welches die Interaktion mit dem Gerät ermöglicht. Das Telefon zeigt den Namen eines Anrufers an und es gibt einen Button zum Anrufen. Der Fernseher zeigt den aktuell eingestellten Sendernamen an und die eingestellte Lautstärke. Der Nutzer kann einen anderen Sender einschalten und die Lautstärke verändern. Der Videorekorder zeigt den Sendernamen an, der eingestellt ist. Zusätzlich zeigt er den aktuellen Status an. Das kann der Status *Aufzeichnen* beziehungsweise *Bereit* sein. Der Home Controller (siehe Unterabschnitt 4.2.1 übernimmt die zentrale Kontrolle aller Geräte. Dazu zeigt er alle Geräte an, die im Haus vorhanden sind, und deren aktuellen Status. Tritt ein Ereignis auf und wird dadurch ein Task erzeugt, dann wird das ebenfalls auf dem Home Controller angezeigt. Dazu werden die dazugehörigen Regeln und der Kontext angezeigt, so dass nachvollzogen werden kann, wann und warum eine Regel ausgelöst wird.

Die Umsetzung jeder Komponente als OSGi-Bundle bedeutet, dass jedes Gerät einen oder mehrere Dienste bereitstellt. So hat beispielsweise der Videorekorder die beiden Dienste *VideoAbspielDienst* und *VideoAufnahmeDienst*. Um seine Dienste bereitzustellen, kann es sein, dass ein Gerät auf andere Dienste zugreifen muss. Dies wird in OSGi als Abhängigkeiten (engl. Dependencies) bezeichnet. Dadurch wird definiert, dass ein Dienst nur ausführbar ist, wenn er auf bestimmte Dienste zugreifen kann.

5.1 Klassifikation von Geräten und Ereignissen

5.1.1 Repräsentation der Konzepte in einer Domänenontologie

Für die semantische Beschreibung der Geräte und ihrer Funktionalität wurden diese in einer Ontologie beschrieben (siehe Abbildung 5.1). Dazu wurde eine Ontologie für typische Geräte, die auch eine älterer Nutzer verwendet, entwickelt. Diese umfasst die Konzepte für typische Geräte aus den Kategorien Kommunikationsgeräte, Unterhaltungsgeräte (Schwarze Ware), elektrische Küche- und Haushaltsgeräte (Weiße Ware) und Steuerungs-

geräte (Heizung, Licht).

Für die Modellierung der Ontologien wurde der Protégé-OWL Editor[1] in der Version 3.3.1 verwendet. In der Geräteontologie wurden alle Geräteklassen, die in dem Prototyp vorkommen, modelliert. Dann wurden die Individuen dieser OWL-Klassen in die Ontologie eingeben. Damit wurden die Verknüpfungen und die Modellierung auf Vollständigkeit überprüft. Die Benennung der OWL-Konzepte und der Attribute ist in Englisch, da das die Standardsprache für die Entwicklung von Ontologien ist.

Die Ontologie ist folgendermaßen aufgebaut. Das oberste Konzept ist das abstrakte Gerät. Das Konzept Gerät hat die Eigenschaften (engl. Properties) hasCapabilities, hasEvent, hasStatus, inAwarenessOf, isLocatedIn und ownedBy. Das sind sogenannte Object-Properties, dass heißt sie haben als Wert ein Objekt, sind mit einem anderen Objekt verbunden. Mit den Objekt-Attributen wird die Relation zwischen den Konzepten definiert. Im Gegensatz dazu haben Datatype-Properties nur einfache Datentypen als mögliche Werte. Die Bedeutung der Object-Properties ist wie folgt:

- `hasCapabilities` ein Gerät besitzt multiple Fähigkeiten (engl. Capabilities pl)
- `hasEvent` ein Gerät hat multiple Ereignisse, die es auslösen kann
- `hasStatus` ein Gerät hat einen Statustypen aus dem Status-Konzept
- `inAwarenessOf` damit wird modelliert, welche Person das Gerät bewußt wahrnimmt
- `isLocatedIn` ein Gerät hat einen Standort, an dem es sich befindet
- `ownedBy` gibt an, welcher Person das Gerät gehört

Die Gerätekategorien (siehe oben) werden als Unterkonzepte von Gerät dargestellt. Somit wird die gewohnte Kategorisierung aus der Realwelt entsprechend durch die Ontologie beschrieben.

Bisher wurde Gerätefunktionalität in anderen Ansätzen per Interfaces definiert. In diesem Ansatz wird die Funktionalität auf einem abstrakten Niveau mit OWL definiert. Es muss dann eine Abbildung von OWL auf die Geräte definiert werden. Durch die Objekt-Eigenschaft wird definiert welche Fähigkeiten ein Gerät hat. In Abbildung 5.2 ist die Ontologie dargestellt, welche die Fähigkeiten-Konzepte definiert.

Ein Fernseher hat die Capabilites VideoRenderer, AudioRenderer und TextRenderer. Der Videorekorder ist mit den Capabilities VideoProvider und VideoCapturer verbunden. Die Konzepte bezeichnen die Funktionalität des Gerätes. Diese Funktionalitäten können von verschiedenen Geräten angeboten werden. Ein Radio ist wie der Fernseher auch die Capabilities AudioRenderer. Die Fähigkeit AudioRenderer ist ein Unterkonzept von Renderer. Daraus ergibt sich eine Hierarchie von Fähigkeiten. Wenn ein Gerät für die Kooperation einen Renderer benötigt, egal welchen, dann kann aus den verfügbaren Renderern einer ausgesucht werden. Das könnte ein AudioRenderer, ein VideoRender oder ein TextRenderer sein.

[1] Protégé ist ein freier, Open-Souce Ontologie-Editor und ein Wissensbasiertes Framework, der vom Stanford Center for Biomedical Informatics Research an der Stanford University School of Medicine entwickelt wurde.
Für mehr Informationen siehe `http://protege.stanford.edu/`.

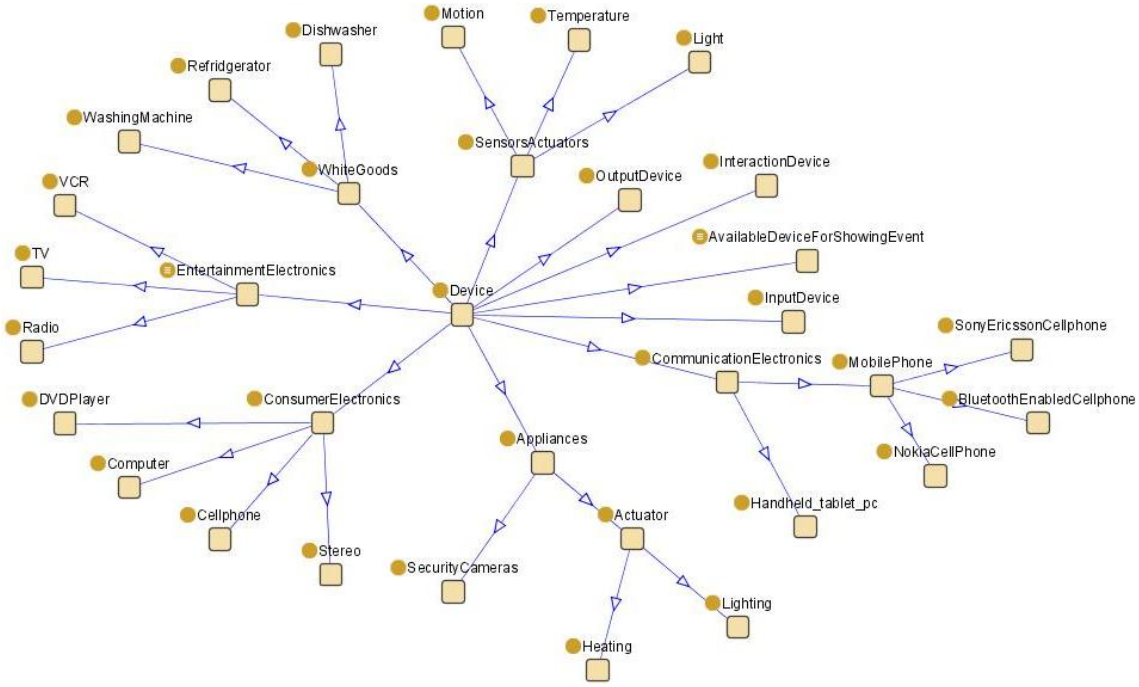

Abbildung 5.1: Geräteontologie

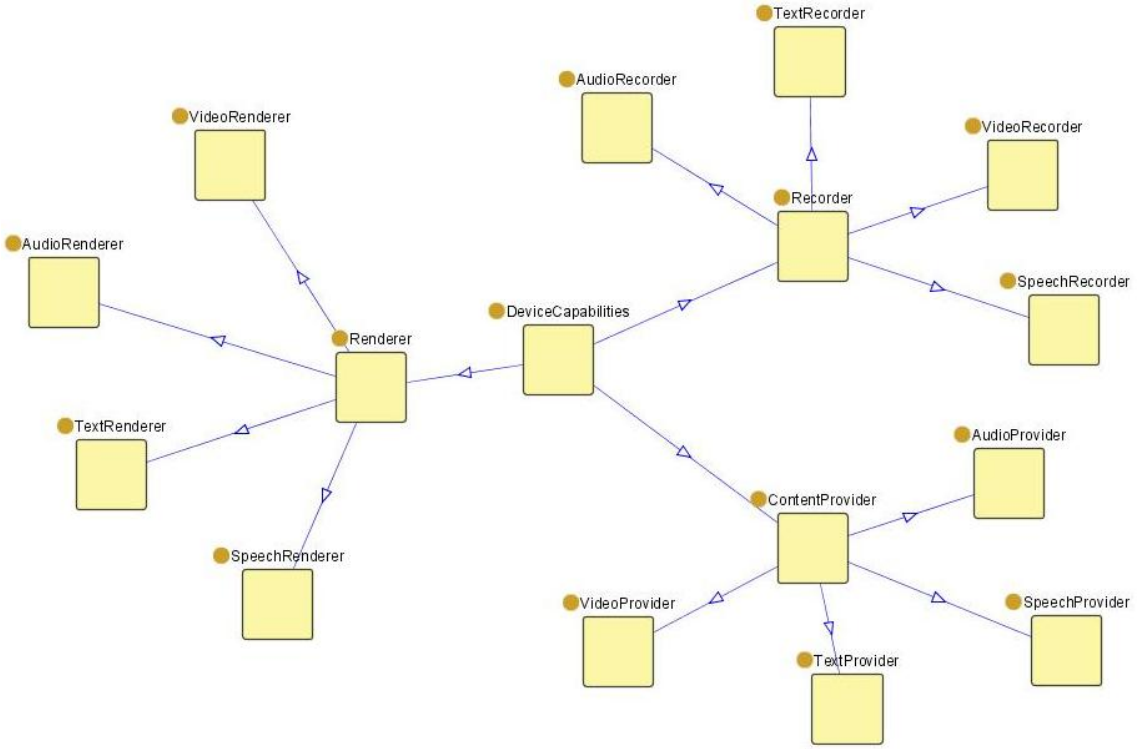

Abbildung 5.2: Gerätefähigkeiten (Device Capabilities)

Somit muss es keine Regel für den Videorekorder und den Fernseher geben, sondern nur für die bestimmten Konzepte, also den Dienst. Dieser Dienst kann von unterschiedlichen Geräten implementiert werden.

Ein VideoRenderer kann auch ein Laptop sein, oder ein VideoBeamer. Natürlich werden für die genaue Spezifikation eines Dienstes Parameter verwendet. So reicht es nicht nur zu wissen, dass ein VideoRenderer im System vorhanden ist. Für die Darstellung eines Videos, sollte der VideoRenderer eine definierte Displaygröße beziehungsweise Videogröße bieten. Die Geräte verwenden unterschiedliche Attribute, um das gleiche Konzept auszudrücken. So hat ein Videobeamer nicht das Attribut Displaygröße sondern die Größe der Videoausgabe wird in Projektionsfläche gemessen. Zwischen diesen Attributen muss es eine semantische Abbildung geben, die diese Attribute gleichsetzt.

5.1.2 Abbilden von OWL-Konzepten auf Dienste

Wie erfolgt die Abbildung von OWL-Konzepten auf Dienste? Die Dienste werden semantisch beschrieben und es gibt eine Abbildung von einer Dienstbeschreibung in der Ontologie auf die vorhandenen Dienste in der Service-Registry. Für die Abbildung wird das Jena Semantic Web Framwork verwendet. Damit lassen sich in Java Ontologien beschreiben. Aus den Java-Klassen können auf direktem Weg OWL-Konzepte erstellt werden.

5.1.3 Kooperation von Geräten

Die Kooperation zwischen Geräten ist generisch und unabhängig von einer bestimmten Anwendungsdomäne. Die Vorbereitung einer möglichen Kooperation erfolgt durch die Integration eines Gerätes in die Middleware.

In der Regel hat die Kooperation einen Ursprung und ein Ziel. Der Ursprung oder die Ursache wird durch ein bestimmtes Ereignis ausgelöst. Das kann zum einen der Nutzer sein, der verschiedene Geräte bedient und eine Aufgabe ausführt um so mit ihnen eine Ziel zu erreichen. Ohne direkte Nutzerinteraktion wird die Kooperation basierend auf einem Regelwerk automatisch gestartet. Interessant ist der Aspekt welche Geräte und mit welcher Funktion an der Kooperation beteiligt werden.

Die Kooperation wird realisiert durch die Orchestrierung der unterschiedlichene Dienste, die von den Geräten angeboten werden. Der Home Controller enthält die Anwendungslogik und ruft die verschiedenen Dienste auf. Damit eine Gerät seinen Dienst oder seine Dienste anbieten kann, muss es auf die Dienste anderer Geräte zugreifen. Damit in diesem Prototypen der Videorekorder das aktuelle Fernsehprogramm für den Nutzer aufnehmen kann, muss der *VideoAufnahmeDienst* mit den Parametern (aktuelles Fernsehprogramm, also Fernsehsender und Dauer der Aufnahme) aufgerufen werden. Die Parameter werden durch den Home Controller ermittelt. Dieser erhält die Werte vom *FernseherDienst*.

Für die Beschreibung der möglichen Kooperationen zwischen Geräten gibt es die sogenannten Tasks. Diese definieren wie verschiedene Geräte miteinander kooperieren können. In einem Task wird definiert welche Gerätetypen beteiligt sind und welche Parameter und Daten von den Geräten benötigt werden (siehe dazu Unterabschnitt 4.1.3). Der Home Controller speichert dieses Wissen, über die möglichen Kooperationen, in der Taskontologie ab.

Damit kann der Home Controller Tasks automatisch finden, wenn dies in einer bestimmten Situation erforderlich ist.

5.1.4 Integration von existierenden Geräten

Bevor Geräte kooperieren können, müssen sie zuerst in die Middleware integriert werden. Die Anforderung I (siehe Abschnitt 2.3) schreibt vor, dass die Integration heterogener Geräte nahtlos erfolgen soll. In der Konzeption (siehe Abschnitt 4.3 wurde festgelegt, dass Geräte in einer Geräteontologie beschrieben werden, was die Integration und das Hinzufügen neuer Geräte ermöglicht.

Im Prototypen wird ein Gerät in die Middleware integriert, durch Starten des entsprechenden Dienstes im Home Controller. Die Device Registry übernimmt die Registrierung des Dienstes und die Device Classification führt die Instanziierung der entsprechenden OWL-Konzepte durch.

5.1.5 Integration neuer Geräte

Die Integration neuer Geräte erfolgt ähnlich der existierender Geräte. Allerdings gibt es für neue Geräte noch keine OWL-Repräsentation. Anhand der semantischen Dienstebeschreibung und mit Hilfe der Device Classification wird ein neues OWL-Konzept in die Ontologie integriert und dann wird das neue Gerät als dessen Instanz hinzugefügt. Bei Konflikten muss der Administrator benachrichtigt werden.

Als neue Geräte werden hier ein Laptop, ein Handy und ein Radio hinzugefügt. Diese haben eine Beschreibung ihrer Dienste und es soll untersucht werden, ob und wie gut die automatische Klassifizierung funktioniert. Der Laptop bietet Funktionalitäten die mit dem Fernseher und dem Videorekorder identisch sind. Mit dem Laptop kann auch Video von einer Quelle aufgezeichnet und vom Laptop abgespielt werden. Das Radio bietet einen AudioWiedergabeDienst, den auch der Fernseher bietet. Das Handy hat die gleiche Funktionalität wie das stationäre Telefon und wird automatisch in die Kategorie Telefon eingefügt.

Zur Verifikation wird der Versuch unternommen ein Gerät zu integrieren, welches in keine der bekannten Kategorien einsortiert werden kann. Die Integration muss in diesem Fall fehlschlagen und der Nutzer wird darüber informiert, dass dieses Gerät manuell in eine Kategorie eingefügt werden muss.

5.1.6 Dienste

Jedes Gerät in diesem Ansatz soll als Dienst (OSGi-Bundle) umgesetzt werden und hat seine spezifische gekapselte Funktionalität, auf die mit Getter- und Setter-Methoden zugegriffen wird.

Ein Dienst ist zum Beispiel der ShowTextService. Dieser wird von der Waschmaschine, vom Geschirrspüler und Fernseher umgesetzt, welche die Fähigkeit (engl. Capability) ShowText haben. Die Dienste ergeben sich aus den Geräten und deren Fähigkeiten.

Suchen eines geeigneten Dienstes

Eine Komponente, die einen Dienst sucht, findet den geeigneten Dienst anhand von semantischen Beschreibungen.

Gibt es mehrere Dienste mit gleicher Funktionalität, dann muss anhand von zu definierenden Kriterien ausgesucht werden, welcher von den verfügbaren Diensten verwendet wird.

In diesem Prototyp gibt es nur einen verfügbaren VideoAufnahmeDienst. Dieser wird gestartet und nimmt die aktuelle Sendung auf. Wenn keine Fehlermeldung eintrifft, geht der Home Controller davon aus, dass der Dienst erfolgreich ausgeführt wird. Der VideoAufnahmeDienst nimmt auf und der Videorekorder, auf dem dieser Dienst läuft, meldet seinen neuen Status, dass er gerade Inhalt aufzeichnet an den Home Controller. Sobald Inhalt aufgenommen wurde, wird der VideoWiedergabeDienst auf dem Videorekorder aktiv und kann von Regeln berücksichtigt werden.

Wenn ein Task ausgeführt wird, dann wird das auf einem Gerät angezeigt, wenn es das unterstützt. Dem Nutzer wird auf dem Fernseher angezeigt, dass dieser Task ausgeführt wurde und das Fernsehprogramm aufgezeichnet wird. Wenn das Telefonat beendet ist, sendet das Telefon ein Event an den Home Controller und daraufhin wird dem Nutzer auf dem Fernseher der Task VideoVonAufnahmeAbspielen angezeigt. Der Task entsteht dadurch, dass die Konzepte in der Ontologie instanziiert werden. Der Reasoner schlussfolgert darauf und leitet diese ab. In dem Moment ab dem der Videorekorder die Sendung aufnimmt, steht Inhalt zum Abspielen zur Verfügung. Der Videorekorder hat sich beim Home Controller angemeldet und den VideoWiedergabeDienst registriert. Statusänderungen eines Gerätes werden dem Home Controller als Events mitgeteilt. Aus dem Status eines Gerätes ermittelt der Home Controller mögliche Tasks. Bei dem Videorekorder, der Zugriff auf Inhalte hat, ist das der Task VideoAbspielen. Der Videorekorder führt den Dienst nur aus und hat keine Informationen darüber wer die Aufnahme gestartet hat. Da der Home Controller den VideoAufnahmeDienst beauftragt hat, das Fernsehprogramm für einen Nutzer aufzunehmen, hat der Home Controller Informationen darüber, welchem Nutzer diese zugeordnet ist und bietet dann diesem Nutzer den Task zum Videoabspielen an. In einem Befehlsspeicher werden gestartete Dienste mit den Benutzerinformationen abgespeichert. Damit der Task auf dem Fernseher angezeigt werden kann, muss der ZeigeInteraktionDienst, des Fernsehers, mit dem Parameter Task aufgerufen werden. Dieser Dienst dient der Interaktion mit einem Gerät. Es wird eine Auswahlliste mit möglichen Tasks auf dem Fernseher dargestellt und daraus wird eine Auswahl getroffen. Der Fernseher dient nur zur Anzeige und Auswahl des Tasks und muss die Semantik nicht verstehen. Das Resultat der Auswahl wird zurück an den Home Controller geschickt und dieser führt den entsprechenden Task aus. In diesem Szenario hat sich der Nutzer dazu entschieden das Fernsehprogramm an der Stelle fortzusetzen, also muss das Video vom Videorekorder abgespielt werden. Dem Videorekorder wird der Befehl zum Abspielen des Videos gesendet mit der Ausgabe auf den Fernseher.

In der Beschreibung eines Dienstes wird festgelegt, wie viele Instanzen dieses Dienstes zur gleichen Zeit ausgeführt werden können. So kann der VideoAufnahmeDienst nur von einer Videoquelle zur gleichen Zeit aufzeichnen.

5.2 Regelschicht

Regeln werden mit der Semantic Web Rule Language (SWRL) definiert. SWRL kombinierte OWL-Konstrukte und Teile der RuleML-Sprache. Regeln sind dazu da zu definieren wie ein Task gebildet wird. Sie arbeiten auf Konzepten aus den Ontologien. Eine Regel kann nur ausgelöst werden (feuern) wenn jede ihrer Bedingungen erfüllt ist. Zu jedem Konzept, dass in der Regel vorkommt, muss es mindestens eine Instanz geben um die Regeln zu erfüllen. Ein Konzept für einen Dienst wird instanziiert, wenn dieser Dienst aktiv ist.

Bei der sequentiellen Abarbeitung von Regeln sollten keine Konflikte zwischen den einzelnen Regeln auftreten. Falls das nicht der Fall ist, dann müssen ab diesem Zeitpunkt die benötigten Dienste reserviert werden, damit keine andere Regel, die benötigten Dienste/Ressourcen verwendet (Auftreten von Race Conditions).

Vorgang des Einlesen und der Verarbeitung der Regeln:

1. Laden der SWRL Regeln aus der SWRL Rule Base und OWL Knowledge Base in die Rule Engine (diese entspricht Java Rule Engine API (JSR94)).
2. Ausführen dieser Regeln auf der Knowlege Base. Prüfung der Resultate.
3. Speichern der Resultate in der OWL Knowledge Base.

Beispiele für SWRL-Regeln mit Vorbedingungen und einer Konsequenz.

> Elternteil(?x,?y) $\wedge$ Bruder(?y,?z) $\Rightarrow$ Onkel (?x, ?z)
>
> Anruf(?x) $\wedge$ Person(?y) $\wedge$ hatAktivität(?y,schautTV) $\wedge$ Videorekorder(?z) $\Rightarrow$ hatStatus(?z, aktiv)

Eine Beispielregel zur Demonstration eines Ereignis, das Interaktion benötigt: Haushaltsgerät (Weiße Ware) und Status=Fehler (Interaktionsereignis) $\Rightarrow$ ZeigeEreignisAufInteraktionsGerätAn

Regelauswertung mit einer Rule Engine

Die Rule Engine hat nun Zugriff auf das Ereignis und den aktuellen Kontext, der die Situation der Anwendungsdomäne beschreibt. Durch dieses Ereignis und den entsprechenden Kontext wird die Regel ausgelöst, die definiert, dass bei einer Unterbrechung (Anruf-Ereignis steht in Ereignis-Ontologie unter Unterbrechung) und aktueller Aktivität des Nutzers (schaut TV) das aktuelle Fernsehprogramm auf einem Videoaufnahme-Gerät aufgezeichnet werden soll. Wenn diese Regel ausgelöst wird, dann müssen die benötigten Parameter für den auszuführenden Task VideoAufnahmeVonFernseher ermittelt werden. Dazu werden vom Fernseher, der aktuelle Fernsehsender ausgelesen. Dieser Fernsehsender wird vom Videoaufnahme-Gerät aufgezeichnet, bis der Nutzer ohne Unterbrechung weiterschauen kann und möchte.

Die Komponente Rule Engine wird als Dienst in OSGi implementiert und beim Home Controller registriert. Sie greift auf den Dienst Knowledge Base zu, um Konzepte zu holen, die für das logische Schlußfolgern benötigt werden. Für die Rule Engine Implementierung wird JESS (Java Expert System Shell) verwendet. JESS ist eine Java-basierte Rule Engine,

die OWL Ontologien verarbeitet. Weil JESS wie auch OSGi Java-basiert ist, wird aus diesem Grund JESS für die Implementierung verwendet.

Die Regeln sind in der SWRL beschrieben und werden in der Knowledge Base abgespeichert. Die SWRL-Jess Bridge ermöglicht die Interaktion zwischen der Knowledge Base, welche die SWRL Regeln und die OWL Ontologien enthält und der Rule Engine, die diese Daten verarbeitet. Die Regeln sind immer in der Rule Engine geladen. Der Arbeitsspeicher dieser Rule Engine enthält die Fakten. Sogenannte Shadow facts sind verbunden mit Java Objekten und sie erlauben es auf Java-Objekten zu schlussfolgern, die sich außerhalb des Arbeitsspeichers befinden. Ein Fakt besteht aus einem Template (vgl. Java Klasse) mit einem Namen, einer Menge von Slots. Das sind einfache Java-Klassen, die nur Getter- und Setter-Methoden enthalten und entspricht der Java Beans Konvention. Die Regeln arbeiten auf diesen Plain-Old-Java-Objects (POJOs).

5.3 Task

Der hier gewählte Ansatz ist Task-orientiert. Mit Tasks (siehe Abbildung 5.3) werden mögliche Aufgaben bezeichnet, die der Nutzer eines Smarthomes ausführen kann. Diese ergeben sich aus dem aktuellen Kontext, der die Gerätekonfiguration, deren Status, die Situation in der Wohnumgebung (möglicherweise auch außerhalb davon) und die Wünsche des Nutzers berücksichtigen. Dem Nutzer werden nur die Tasks angeboten, die zu dem aktuellen Zeitpunkt möglich sind. Dies ist notwendig, da das System dem Nutzer das Leben erleichtern soll. Dieser Automatismus kann den Nutzer irritieren. Zu einem Zeitpunkt wird eine Sendung aufgezeichnet wenn jemand anruft. Beim nächsten Mal nimmt der Videorekorder gerade eine andere Sendung auf und der Tasks VideoaAufnehmen wird nicht mehr angeboten.

Im Folgenden wird beschrieben, wie die möglichen Tasks ermittelt werden. Ein Task ist die rechte Seite einer Regel der Form (Bedingung $\Rightarrow$ Task). Jedem Task ist ein Subjekt (hasSubject Subject (Content)) und eine Ressource (requiresResource (Device)) zugeordnet. Das Subjekt ist der Inhalt, der dargestellt werden soll oder von dem eine Aufgabe ausgeht. Die Ressource ist ein Gerät, welches benötigt wird um den Task auszuführen.

Anhand des folgenden Beispiels werden die Bestandteile eines Task gezeigt. *Der Hausbewohner sitzt vor dem Fernseher und schaut sich eine interessante Sendung an. Jemand ruft an. Falls Videoaufnahmegerät aktiv ist, wird das aktuelle TV-Programm von diesem Gerät aufgezeichnet.*

```
Task: TVProgrammAufVideoRekorderAufnehmen
Bedingung: Anruf und (Person sieht fern) und (Videorekorder=aktiv)
Anruf = Telefon klingelt (Kontextdienst erhält den Wert Telefon=wird angerufen)
PersonSiehtFern = Person und Aktivität (sitzt vor TV und TV=aktiv)
AufnahmeVideoQuelle = aufnehmen(Video) von Quelle(Video)
benötigt {VideoQuelle, VideoAufnahmeGerät}
```

Sobald die Sendung aufgezeichnet wurde steht ein neuer Task *VideoAbspielen* zur Verfügung. Nach dem Telefonat wird dem Hausbewohner auf dem Fernseher die Möglichkeit angeboten, die aufgezeichnete Sendung abzuspielen. Dieser Punkt lässt sich noch weiter differenzieren. Der Task kann auch unabhängig von dem Anruf entstehen. Wenn Videoinhalte zur Verfügung stehen, dann wird dem Nutzer angeboten sich das Video anzuschauen.

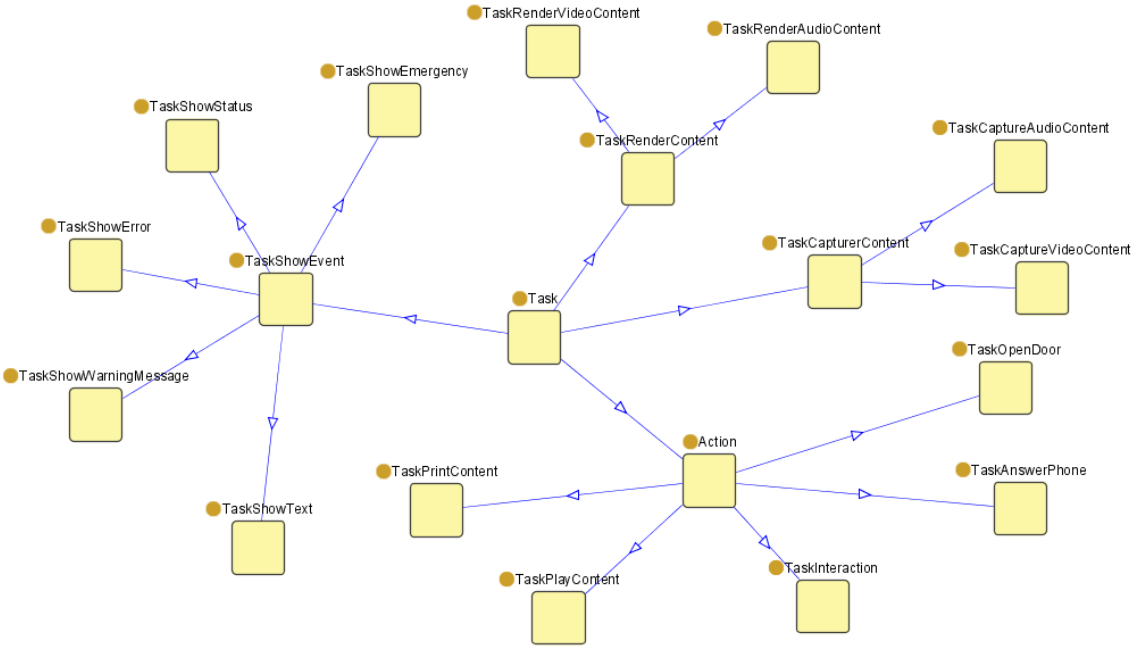

Abbildung 5.3: Ausschnitt aus der Task-Ontologie

Dieser Aspekt ist noch abhängig von den Nutzerpräferenzen für eine solche Situation und anderen Parameter wie der Tageszeit. Ein Task enthält die Aktionen, also die Logik, die in einer bestimmten Situation ausgeführt werden soll (Task $\Longrightarrow$*Aktionen).

Für die Erstellung eines Tasks müssen dessen Bedingungen erfüllt werden. In diesen Beispielen werden Instanzen von einer VideoQuelle (Videorekorder) und einem VideoRenderer (TV) benötigt und diese müssen den Zustand aktiv haben.

Wenn mehrere Instanzen eines Konzeptes vorhanden sind, muss die Middleware ermitteln, welches Gerät das Beste für diese Situation ist. In dem gewählten Beispiel gibt es mehrere Videorekorder und auf welcher wird für eine Videoaufnahme verwendet? Die Auswahl erfolgt anhand von Qualitätseigenschaften, zeitlicher Verfügbarkeit und Prioritäten und wird von der Komponente Semantic Service Discovery (siehe Unterunterabschnitt 4.2.14.3) durchgeführt.

T-BOX (Konzepte) $\Longleftrightarrow$ A-BOX (Instanzen)

In Beschreibungslogiken werden Konzepte mit T-BOX (Terminological axioms) bezeichnet. Die konkrete Situation, also die Instanzen, die den Zustand der modellierten Welt repräsentieren werden mit A-BOX bezeichnet.T-Box: *Jeder Angestellte ist eine Person.* Wird benutzt für die Klassifizierung. A-Box: *Bob ist ein Angestellter.* Für die Instanz Bob kann überprüft werden von welchem Konzept sie ist (instance checking). Darstellung von den Konzepten T-Box und den Individuen der A-Box in einem Beispiel.

```
Task: VideoAbspielen = benötigt VideoQuelle und VideoRenderer
VideoQuelle (Videorekorder liefert Videosignal)
VideoRenderer (TV, der Video anzeigen kann)
```

5.4 Geräte als OSGi-Bundles

Die Device-Registry ist die zentrale Stelle, in der Geräte registriert werden. Ein aktives Gerät meldet sich bei dem Home Controller an und wird in der Device Registry registriert. In OSGi ist das die ServiceRegistry. Diese verwaltet alle Zustände eines Dienstes. Ein Bundle kann den Zustand aktiv, inaktiv, haben. Es kann installiert, aktualisiert (update) und deinstalliert werden.

Ein Gerät wird duch seine Dienstebeschreibung in der Manifest-Datei spezifiziert. In dieser wird genau definiert, welche Funktion der Dienst hat und welche anderen Bundles von ihm benötigt werden. Das Bundle kann erst gestartet werden, wenn alle benötigten Bundles verfügbar sind. Beim Aktualisieren eines Bundles wird geprüft, ob es Abhängigkeiten zu anderen Bundles gibt und diese müssen dann erst beendet werden.

5.4.1 Ereignisbehandlung

Geräte registrieren sich beim Home Controller für Ereignisse, die für sie relevant sind. Wenn das Ereignis auftritt informiert der Home Controller alle Geräte, die als Listener registriert sind. Der Home Controller registriert für alle Geräteereignisse Service-Listener im Framework. Er ist somit der zentrale Ereignisvermittler zwischen den verschiedenen Geräten. Die Rule Engine ist für alle Ereignisse als Listener eingetragen und wird vom Home Controller über alle Ereignisse informiert. Die Ereignisbehandlung erfolgt in der Methode des jeweiligen Listeners.

Ein Anrufer ruft an und das Telefon informiert den Home Controller über dieses Ereignis. Der Home Controller leitet das Ereignis an die Rule Engine weiter. Bei diesem Ereignis soll der Nutzer informiert werden, um auf das Ereignis zu reagieren. Der Home Controller befragt den Kontextdienst nach dem aktuellen Aufenthaltsort des Nutzers, wenn der aktuelle nicht bekannt ist. Als Antwort erhält den Aufenthaltsort des Nutzers. Der befindet sich gerade vor dem Fernseher. Der Fernseher ist geeignet um das Ereignis anzuzeigen. Also wird der Nutzer auf dem Fernseher über das Ereignis informiert.

Die Ereignisbehandlung wird im Prototypen mit dem OSGi-Event Admin Service realisiert. Dieser verteilt Ereignisse an Listener. In OSGi werden Ereignisse so definiert, das sie einen Typen und eine Nachricht enthalten. Darauf setzt das hier entwickelte Ereignismodell auf.

5.4.2 Kontextbenachrichtigung

Jede Kontextänderung die auftritt löst ein Ereignis aus, worüber die Abonnenten diesen Kontexts informiert werden. Die Einbindung von Kontext kann über den BIB3R-Kontextdienst erfolgen. In diesem Prototypen werden allerdings keine Kontextdaten erfasst. Sie werden durch Nutzereingaben, also Ändern von Attributen in der GUI simuliert. Die Rule Engine abonniert alle Kontextereignisse und löst Regeln aus, wenn die Bedingungen der Regeln an die Kontextdaten geknüpft sind.

In diesem Kapitel wurden der Prototyp und die bei der Implementierung betrachteten Aspekte vorgestellt.

6 Bewertung der Arbeit

Die nahtlose Integration und Kooperation von Geräten als eine Voraussetzung für das Ubiquitous Computing ermöglicht dem Nutzer einen intuitiveren Umgang mit seiner technischen Umgebung und schafft ihm benötigte Freiräume. Die dazu in dieser Arbeit entworfene Lösung wird nachstehend bewertet.

Umsetzung der aufgestellten Anforderungen

Die in der Anforderungsanalyse (siehe Kapitel 2) aufgestellten funktionalen Anforderungen konnten größtenteils umgesetzt werden. Folgende Anforderungen, die über die reine Middleware-Lösung hinausgehen, wurden noch nicht umgesetzt. Das sind die Anforderungen XI (*Das System muss die geeignete Modalität zur Ausgabe und Eingabe von Informationen auswählen*), XIII (*Das System soll es ermöglichen, dass ein Gerät einen Fehlerbericht an den Händler sendet*), XIV (*Dem Nutzer sollen nur Aufgaben angezeigt werden, die seiner Intention und seinem Kontext entsprechen*), XVI (*Das System soll den Nutzer an Tätigkeiten und Termine erinnern können*), XVII (*Weiterführung der Bedienung auf einem anderen Gerät (Task Continuation)*) und XIX (*Protokollierung aller Nutzerinteraktionen mit den Geräten*). Damit diese Anforderungen umgesetzt werden können, bedarf es weiterer Komponenten in der Systemarchitektur.

Es gibt für die verschiedenen Geräteklassen eine semantische Repräsentation in der Geräteontologie und es wurden verschiedene Geräte im Prototypen als Dienst implementiert. Durch die Nutzung von Ontologien zur semantischen Beschreibung der Domäne und der Kontextdaten wird eine abstrakte Beschreibung geweählt und dadurch ist die Integration von vielfältigen Systemen möglich. Damit werden die Anforderungen I (*Integration heterogener Geräte*), II (*Zwischen Geräten soll eine nahtlose Kooperation, ohne vorherige Konfiguration, möglich sein*), III (*Geräte müssen eine semantische Selbstbeschreibung haben* und IV (*Finden von Geräten mit Suche nach deren Fähigkeiten*)erfüllt.

Die nichtfunktionalen Anforderungen, wie Usability, Sicherheitsanforderungen, Leistung und Effizienz wurden in dieser Arbeit nicht konkret behandelt. Denn diese erfordern ein ausgiebiges Testen durch eine ausgewählte Nutzergruppe und die Integration der Middleware in eine Smarthome-Umgebung. Die Tests können nur unter realen Bedingungen, am Besten in einem richtigen Smarthome durchgeführt werden. Auf Grundlage dieser theoretischen Überlegungen wurde ein erster Prototyp entwickelt, der sich durch seine Anwenderfreundlichkeit auszeichnet.

Hilfestellung in typischen Situationen?

Werden die in der Aufgabenstellung genannten typischen Situationen von älteren Menschen erfasst und kann den Menschen in diesen Situationen geholfen werden?

Darstellung von Funktionalität auf leicht verständliche Weise und Anbieten von verfügba-

ren Aufgabe. Es steht nicht das einzelne Gerät im Vordergrund, sondern das Zusammenspiel der unterschiedlichen kooperierenden Geräte ist entscheidend. Das intelligente System ermittelt automatisch anhand der Situation des Nutzers und der Gerätekonfiguration welche Aufgaben mit den Geräten möglich sind. Daraufhin werden ihm nur diese Aufgaben angezeigt, aus denen er dann die gewünscht auswählen und ausführen kann. Somit ist die Interaktion mit Geräten schon sehr vereinfacht. Der Nutzer muss nicht mehr überlegen, welche Geräte er alle einstellen und wie konfigurieren muss. Darüber hinaus, kann der Nutzer so auch Aufgaben mit Geräten ausführen, die ihm vorher nicht bewusst waren. Viele Geräte verstecken eine Vielzahl von Funktionalität hinter unzähligen Menüs, die ohne ausgiebiges Studium der Handbücher oder Hilfen nicht erreicht werden können. Besonders ältere Menschen sind sehr vorsichtig im Umgang mit Technik und möchten nichts falsch machen.

Inwieweit unterstützt das System ältere Menschen in deren typischen Situationen? Notfälle werden erkannt durch Sensoren, die in den Haushalt integriert werden. Diese überwachen den Gesundheitszustand des Nutzers und alarmieren Rettungskräfte wenn ein kritischer Zustand erreicht ist. In diesem System werden Regeln definiert, mit denen sich beliebige Situationen definieren und kontrollieren lassen. Mit diesen Regeln und der Auswertung von Kontextdaten wird auf Notfälle reagiert.

Vergesslichkeit: Das System bietet dem Nutzer in jeder Situation die möglichen Funktionalität an und berücksichtigt seine. Da viele Vorgänge automatisiert werden, muss der Nutzer sich nicht mehr an alles erinnern. Das birgt andererseits die Gefahr, dass der Nutzer durch diese Entlastung weniger gefordert wird und die Vergesslichkeit zunimmt. Hilflosigkeit im Umgang mit fehlerhaften Arbeitsbedingungen von Geräten und Anwendungen. Dem Nutzer werden leicht verständliche Funktionen dargestellt. Da die Geräte miteinander kooperieren, entfallen für den Nutzer viele Situationen in denen Fehler auftreten können. Im Fehlerfall wird dem Nutzer ein Lösungsvorschlag angeboten, so dass keine Gefühl der Hilflosigkeit mehr auftreten sollte.

Wie lässt sich messen, dass das entwickelte System intuitiver als vergleichbare Systeme ist? Das System ist intuitiver, wenn es dem Nutzer den Umgang mit seiner elektronischen Umgebung erleichtert.

Offene Fragen

- Untersuchung der Usability. Wie einfach lässt sich das System vom Nutzer bedienen? Welche Voraussetzungen benötigt der Nutzer? Muss er ein Fachmann sein? Wie geht ein Laie damit um? Gibt es Unterscheide zwischen jungen und älteren Nutzern? Wird Fachwissen benötigt?
- Wie ist die Performanz? Erfolgt eine Antwort des Systems in Echtzeit oder wird ein Aktion Film erst aufgenommen, nachdem Telefonat beendet ist?
- Wie kann der Nutzer eigene Regeln konfigurieren/erfassen?
- Nutzerverhalten versus erwartetes Systemverhalten.

7 Zusammenfassung

Das Ziel dieser Diplomarbeit war es eine Middleware zu entwickeln, die die nahtlose Integration und Kooperation von Geräten ermöglicht. Dadurch soll im Umfeld des Ubiquitous Computing der Umgang mit der Technik für den Nutzer intuitiver werden. Im Zusammenhang mit der Context-Awareness kann insbesondere älteren Menschen ein unabhängigeres Leben ermöglicht werden.

In Kapitel 2 werden Anwendungsbereiche in der Wohnumgebung identifiziert, die typische Situationen älterer Menschen, wie Notfälle, Vergesslichkeit und Verzweiflung im Umgang mit fehlerhafter Technik, vorstellen. Besonders in diesen Situationen soll die entwickelte Middleware helfen. Zuerst werden Szenarien beschrieben, die ein realistisches Bild vom Einsatzbereich vermitteln. Aus diesen werden Anwendungsfälle abgeleitet, die Anforderungen für die Konzeption und die spätere Umsetzung aufstellen. Zunächst wurden verwandte Arbeiten und existierende Technologien auf diese Anforderungen untersucht (siehe Kapitel 3). Die daraus gewonnenen Erkenntnisse flossen direkt in den Entwurf der konzeptionellen Architektur ein (siehe Abschnitt 4.2).

Desweiteren wurde der Einsatz von Ontologien, zur Modellierung der Smart Environments, untersucht. Für die Modellierung vielfältiger Anwendungsdomänen eignen sich Ontologien, die mit der Web Ontology Language (OWL) beschrieben werden, sehr gut. OWL bietet den Vorteil der umfassenden semantischen Beschreibung von Konzepten und der Interoperabilität zwischen den Ontologien. Deshalb sind OWL-Ontologien die erste Wahl bei Projekten mit dem Ziel der Entwicklung von Smart Environments und kontextbewußten Infrastrukturen.

Regeln und Policen für die nahtlose Kooperation und Integration von Geräten zu finden, war ein weiterer Bearbeitungsschwerpunkt. Diesbezüglich wurde in der Konzeption eine dafür entwickelte „Dreischichtige Kooperationsarchitektur“ eingeführt (siehe Abschnitt 4.1. Sie definiert die Policen auf der obersten, Regeln auf der mittleren und Tasks auf der untersten Schicht. Es wird erläutert, wie sich diese Schichten voneinander unterscheiden, wie sie erstellt und für die Kooperation und Integration verwendet werden.

Parallel zu der Konzeptionsphase wurde mit der Erstellung eines Prototypen begonnen. In diesem wurden Konzepte implementiert um diese zu verifizieren. So konnten bestimmte Probleme verdeutlicht werden. Desweiteren zeigt der Prototyp die Realisierbarkeit des gewählten Ansatzes. Er basiert auf OSGi. Jedes Gerät, welches in die Middleware integriert wird, ist als Dienst, also als OSGi-Bundle realisiert. Die Idee zur Verwendung von OSGi, ist während der Sichtung der verwandten Arbeiten entstanden. Es wird bereits in Smarthomes, für die Vernetzung von Haushaltsgeräten verwendet und in großen Forschungsprojekten eingesetzt. Der Prototyp erlaubt es allerdings nicht, Regeln mit einem Rule Builder automatisch erstellen zu lassen. Sie müssen manuell in der Regelsprache SWRL beschrieben werden. Es wurde auch keine Anbindung an einen Kontextdienst implementiert, da dies keinen weiteren Nutzen für die Verifikation des hier gewünschten Ansatzes gebrachtet

hätte. Kontextdaten werden im Protoyp durch Nutzereingaben simuliert. Bisher wurde der Prototyp nur auf einem PC getestet. Das bedeutet alle OSGi-Bundles laufen in der freien OSGi-Implementierung „Knopflerfish“ auf einer virtuellen Maschine. Als nächstes muss getestet werden, wie OSGi-Bundles auf heterogenen Hardware-Plattformen installiert werden können und in einem solchen verteilten System lauffähig sind. Dabei müssen die Aspekte zur verteilten Kommunikation in OSGi und die Performance eines solchen Systems untersucht werden. Dies wurde vorliegend nicht untersucht, da es dem Schwerpunkt der Aufgabenstellung nicht entsprach.

Es konnte nur theoretisch untersucht werden, ob die dynamische und nahtlose Integration und Kooperation dazu beiträgt den Umgang mit Technik intuitiver zu gestalten. Die Annahme, dass besonders älteren Menschen dadurch ein erleichtertes und unabhängigeres Leben ermöglicht werden kann, kann nur unter Vorbehalt bestätigt werden. Es ist sicherlich eine große Hilfe wenn die Geräte automatisch für den Nutzer Aufgaben erledigen, ihn in allen erdenklichen Situationen unterstützen und im Fehlerfall Hilfe organisieren. Das kann aber auch dazu führen, dass der Nutzer sich zu sehr an die Unterstützung gewöhnt und dadurch noch abhängiger. Möglicherweise wird ein Nutzer noch vergesslicher, wenn sein Alltagsumfeld vollständig automatisiert ist und ihn auch an jeden Termin erinnert. Außerdem ist abzusehen, dass ein solches System nicht in jeder Situation die korrekte Entscheidung treffen kann (vgl. [29]). Das wirft wiederum die Frage auf, ob der Nutzer irritiert wird, wenn er nicht mehr jede Entscheidung nachvollziehen kann.

Ein weiterer Aspekt, der nicht betrachtet wurde, ist der der Privatssphäre. In diesem Bereich ist noch viel Forschungsarbeit erforderlich, damit solche Lösungen in der Praxis eingesetzt und von der Nutzergruppe akzeptiert werden.

Momentan gibt es mehrere große EU-Projekte, die sich mit Smarthomes auseinandersetzen. Es kann davon ausgegangen werden, dass die Mehrheit der Bevölkerung in den nächsten Jahren von der Unterstützung der ubiquitären Technik profitieren wird, wenn diese finanzierbar geworden ist.

Als Ausblick bleibt die Umsetzung der Aufgabe, Policies vollautomatisch zu lernen, in Regeln zu transformieren und auch vollkommen unbekannte Geräte und Ressourcen automatisch in das System zu integrieren, damit sie miteinander kooperieren können. Dabei ist zu gewährleisten, dass der Nutzer jederzeit alle Prozesse im Innern des Systems nachvollziehen und das Systemverhalten beliebig abschalten oder dynamisch ändern kann.

Literaturverzeichnis

[1] Abowd, Gregory D. ; Dey, Anind K. ; Brown, Peter J. ; Davies, Nigel ; Smith, Mark ; Steggles, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999. – ISBN 3–540–66550–1, S. 304–307

[2] Banavar, Guruduth ; Beck, James ; Gluzberg, Eugene ; Munson, Jonathan ; Sussman, Jeremy ; Zukowski, Deborra: *Challenges: an application model for pervasive computing*. Boston, Massachusetts, United States : ACM Press, 2000. – 266–274 S. – ISBN 1–58113–197–6

[3] Brügge, Bernd ; Dutoit, Allan H.: *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java*. Pearson Studium, 2004

[4] Chen, Harry ; Perich, Filip ; Finin, Tim ; Joshi, Anupam: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In: *International Conference on Mobile and Ubiquitous Systems: Networking and Services*. Boston, MA, August 2004

[5] Ding, Y. ; Elting, C. ; Scholz, U.: Seamless integration of output devices in intelligent environments: Infrastructure, strategies and implemeentation. In: *Intelligent Environments, 2006. IE 06. 2nd IET International Conference on* Bd. 1, 2006. – ISBN 0537–9989, S. 21–30

[6] Esler, Mike ; Hightower, Jeffrey ; Anderson, Tom ; Borriello, Gaetano: Next Century Challenges: Data-Centric Networking for Invisible Computing. The Portolano Project at the University of Washington. In: *Mobile Computing and Networking*, 1999, S. 256–262

[7] Euzenat, Jerome: Alignment Infrastructure for Ontology Mediation and other Applications. In: Hepp, Martin (Hrsg.) ; Polleres, Axel (Hrsg.) ; Harmelen, Frank van (Hrsg.) ; Genesereth, Michael R. (Hrsg.): *MEDIATE2005* Bd. 168, CEUR-WS.org, 2005. – online http://CEUR-WS.org/Vol-168/MEDIATE2005-paper6.pdf, S. 81–95

[8] Fensel, Dieter (Hrsg.) ; Lausen, Holger (Hrsg.) ; Polleres, Axel (Hrsg.) ; Bruijn, Jos D. (Hrsg.) ; Stollberg, Michael (Hrsg.) ; Roman, Dumitru (Hrsg.) ; Domingue, John (Hrsg.): *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Heidelberg : Springer-Verlag, 2006. – ISBN 3540345191

[9] Friedman-Hill, Ernest. *Java Expert System Shell (Jess), the Rule Engine for the Java Platform*

[10] Gruber, Thomas R.: A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220. – ISSN 1042–8143

[11] Gu, T. ; Pung, H.K. ; Zhang, D.Q.: Toward an OSGi-based infrastructure for context-aware applications. In: *Pervasive Computing, IEEE* 3 (2004), S. 66–74. – ISSN 1536–1268

[12] Hammerschall, Ulrike: *Verteilte Systeme und Anwendungen - Architekturkonzepte, Standards und Middleware-Technologien.* Pearson Studium, 2005

[13] Hand, Von D. J. ; Mannila, Heikki ; Smyth, Padhraic: *Principles of Data Mining.* The MIT Press, August 2001. – ISBN 026208290X

[14] Horrocks, Ian ; Patel-Schneider, Peter F. ; Boley, Harold ; Tabet, Said ; Grosof, Benjamin ; Dean, Mike: SWRL: A Semantic Web Rule Language Combining OWL and RuleML / World Wide Web Consortium. 2004. – W3C Member Submission

[15] Kokar, Mieczyslaw M.: « Ontology Based Situation Awareness and High Level Fusion: Methods and Tools» - Workshop on Information Fusion (Fusion06). Florence (Italy), july 2006

[16] Magerkurth, C. ; Etter, R. ; Janse, M. ; Kela, J. ; Kocsis, O. ; Ramparany, F.: An intelligent user service architecture for networked home environments. In: *Intelligent Environments, 2006. IE 06. 2nd IET International Conference on* 1 (5-6 July 2006), S. 361–370. – ISSN 0537–9989

[17] McBride, B.: Jena: a semantic Web toolkit. In: *Internet Computing, IEEE* 6 (2002), S. 55–59. – ISSN 1089–7801

[18] Messer, A. ; Kunjithapatham, A. ; Sheshagiri, M. ; Song, H. ; Kumar, P. ; Nguyen, P. ; Yi, Kyoung H.: InterPlay: a middleware for seamless device integration and task orchestration in a networked home. In: *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, 2006, S. 10 pp.

[19] Patel-Schneider, Peter F. ; Hayes, Patrick ; Horrocks, Ian ; Patel-Schneider, Peter F. (Hrsg.) ; Hayes, Patrick (Hrsg.) ; Horrocks, Ian (Hrsg.): OWL Web Ontology Language Semantics and Abstract Syntax / World Wide Web Consortium. 2004. – Forschungsbericht

[20] Ricquebourg, Vincent ; Durand, David ; Menga, David ; Marhic, Bruno ; Delahoche, Laurent ; Loge, Christophe ; Jolly-Desodt, Anne-Marie: Context Inferring in the Smart Home: An SWRL Approach. In: *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on* Bd. 2, 2007, S. 290–295

[21] Sanders, Jane. *GT Research Horizons - Winter 2000*

[22] Satoh, Ichiro: Mobile Agents for Ambient Intelligence. In: *MMAS*, 2004, S. 187–201

[23] Sinderen, M.J. van ; Halteren, A.T. van ; Wegdam, M. ; Meeuwissen, H.B. ; Eertink, E.H.: Supporting context-aware mobile applications: an infrastructure approach. In: *Communications Magazine, IEEE* 44 (2006), Nr. 9, S. 96–104. – ISSN 0163–6804

[24] Sloman, E.: Security and management policy specification. In: *Network, IEEE* 16

(Mar/Apr 2002), Nr. 2, S. 10–19. – ISSN 0890–8044

[25] Stavroulaki ; Demestichas ; Adamopoulou ; Demestichas: Distributed Web-based Management Framework for Ambient Reconfigurable Services in the Intelligent Environment. In: *Mobile Networks and Applications* 11 (2006), Dezember, S. 889–900

[26] Steller, Luke ; Krishnaswamy, Shonali ; Newmarch, Jan: Discovering Relevant Services in Pervasive Environments Using Semantics and Context. In: Mostéfaoui, Soraya K. (Hrsg.) ; Maamar, Zakaria (Hrsg.) ; Giaglis, George M. (Hrsg.): *IWUC*, INSTICC Press, 2006. – ISBN 978–972–8865–51–1, S. 3–12

[27] Wang, Xiao H. ; Zhang, Daqing ; Gu, Tao ; Pung, Hung K.: Ontology Based Context Modeling and Reasoning using OWL. In: *PerCom Workshops*, 2004, S. 18–22

[28] Winkler, Matthias ; Heinrich, Matthias ; Behring, Alexander ; Steinmetz, Joachim ; Dargie, Waltenegus: EMODE - ein Ansatz zur werkzeugunterstützten Modellierung multimodaler, adaptiver Benutzerschnittstellen. In: Koschke, Rainer (Hrsg.) ; Herzog, Otthein (Hrsg.) ; Rödiger, Karl-Heinz (Hrsg.) ; (Hrsg.), Marc R. (Hrsg.): *Informatik 2007 - Band 1.* Bremen, Germany, September 2007 (Lecture Notes in Informatics)

[29] Wustmann, Patrick: *Ein Kontextbasiertes Modell für den situationsabhängigen Einsatz von Ein- und Ausgabemodalitäten in adaptiven Benutzerschnittstellen*, Technische Universität Dresden, Diplomarbeit, December 2006

[30] Zhang, Tao ; Bruegge, Bernd: Empowering the User to Build Smart Home Applications. In: *Toward a Human-friendly Assistive Environment*, 2004

Abbildungsverzeichnis

2.1 Kommunikationsbeziehungen zwischen Anwendungsfalldiagrammen 13

3.1 SOUPA Ontology 2004-06 [4] 30
3.2 The Task Composition Screen 33
3.3 The Session Screen 34
3.4 Überblick über die SOCAM Architektur [11] 38
3.5 CONtext ONtology CONON [11] 40
3.6 Beispiele einer Teilregelmenge basierend auf einer Forward-Chaining Rule Engine 40

4.1 Dreischichtige Kooperationsarchitektur 50
4.2 Architektur 55
4.3 Beispiel eines Entscheidungsbaums 76

5.1 Geräteontologie 81
5.2 Gerätefähigkeiten (Device Capabilities) 81
5.3 Ausschnitt aus der Task-Ontologie 87

Danksagung

Hiermit möchte ich mich bei denen bedanken, die mir geholfen haben, bis hierher zu kommen. In guten Zeiten und in schlechten Zeiten.